Universidade Católica de Santos - UniSantos

Mestrado em Informática

Autorização Integrada entre Portais e Globus baseada no Modelo RBAC

Vanderlei F. da Costa

Santos

2008

Autorização Integrada entre Portais e Globus baseada no Modelo RBAC

Vanderlei F. da Costa

Dissertação apresentada à Reitoria de Pós-Graduação Stricto Sensu da Universidade Católica de Santos, para a obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Hermes Senger
Co-orientador: Prof. Dr. Auri M. Rizzo Vincenzi

Unisantos
Santos, SP, Brasil
2008

Dedico este trabalho aos meus pais pela força em manter-me sempre no caminho correto.

Dados Internacionais de Catalogação
Sistema de Bibliotecas da Universidade Católica de Santos
SibiU

C837a Costa, Vanderlei Ferreira da
Autorização Integrada entre Portais e Globus
baseada no Modelo RBAC / Vanderlei Ferreira da Costa - Santos:
[s.n.] 2008.
135 f. ; 30 cm. (Dissertação de Mestrado - Universidade Católica de
Santos, Programa em Ciência da Computação)

I. Costa, Vanderlei Ferreira da II. Título

CDU 681.3:004 (043.3)

Agradecimentos

- Agradeço a Deus, por ter permitido que eu chegasse até aqui.
- Agradeço especialmente a minha esposa e ao meu filho, por suportarem minha ausência durante esse longo período de estudo.
- Meus sinceros agradecimentos a todos os Mestres pelo privilégio de assistir às suas aulas e especialmente, aos professores Doutor Hermes Senger e Doutor Auri Marcelo Rizzo Vincenzi pela paciência, pelos ensinamentos, pelas conversas e pelos esclarecimentos que contribuíram para o resultado final deste trabalho.
- Agradeço aos colegas, Flavio Jose, Franco Sandonato, Jorge Chiara, Maria Madope, Marcos, Henrique Gagliardi, Paulo Roberto e Silvio Stanzani, que me ajudaram durante toda essa caminhada.
- Agradeço também ao meu gerente Nelson Cardoso pelo apoio recebido durante todo o período do curso e pela confiança em mim depositada.

Resumo

Plataformas de grade computacional têm sido adotadas para promover o compartilhamento, agregação e coordenação de grandes quantidades de recursos geograficamente distribuídos e multi-institucionais. Em tais ambientes, que envolvem grandes quantidades de recursos localizados em diversos domínios administrativos e sujeitos a uma diversidade de políticas de controle de acesso, o controle de acesso é obrigatório. Como principal contribuição, o presente trabalho estende o arcabouço (*framework*) de portais GridSphere com o objetivo de fornecer ferramentas de controle de acesso que podem ser utilizadas para o desenvolvimento de aplicações para grades computacionais. Os mecanismos e ferramentas propostos também realizam o controle de acesso no nível de invocação de serviço, que pode ser usado por qualquer aplicação compatível com OGSA que realize invocações a serviços de grade. Nossa abordagem permite a integração e consistência entre políticas de autorização aplicadas no lado do portal e no lado do provedor de serviço.

Abstract

Grid platforms have been increasingly adopted to promote the sharing, aggregation, and coordination of large amounts of geographically distributed and multi-institutional resources. In such environments, which involve large amounts of resources placed at several administrative domains and subject to multiple access control policies, access control is mandatory. As the main contribution, in this dissertation we extend the GridSphere portal framework, aiming to provide tools and mechanisms for access control which can be used for the development of portal-based grid applications. The mechanisms and tools proposed also enforce access control at the service invocation level, which can be used by virtually any OGSA compliant application which invokes grid services. Our approach allows for the integration and consistency between authorization policies applied both at portal side and to the service provider side.

Sumário

Resumo **iv**

Abstract **v**

1 Introdução **1**
1.1 Objetivos 5
1.2 Metodologia 5
1.3 Organização da Dissertação 5

2 Segurança de Sistemas **7**
2.1 Criptografia 7
2.2 Assinatura Digital 10
2.3 Certificados Digitais 10
2.4 Autoridade Certificadora 14
2.5 Autenticação 16
2.6 Autorização 16
2.7 Integridade 17
2.8 Comunicação Segura 18
2.9 Modelos de Controle de Acesso 19
2.9.1 Modelo DAC 20
2.9.2 Modelo MAC 21
2.9.3 Modelo RBAC 24
2.10 Conclusão 27

3 Grades Computacionais **28**
3.1 Open Grid Services Architecture 30
3.2 Globus Toolkit 4 32
3.2.1 Certificados Proxy 37
3.2.2 Autenticação Única 39
3.2.3 Delegação 39
3.2.4 MyProxy 41
3.2.5 Soluções para a Autorização 43
3.3 Segurança no GT4 50
3.4 Portais de Grade 53
3.4.1 A Ferramenta GridSphere 56
3.4.2 A Ferramenta OGCE 58
3.4.3 A Ferramenta Clarens 61
3.5 Segurança em Portais de Grade 64
3.6 Conclusão 68

4 Proposta IAGP **71**
4.1 Requisitos Funcionais . . . 73
4.2 Visão Geral . . . 76
4.3 Especificação dos Casos de Uso . . . 79
4.4 Aspectos da Implementação . . . 84
4.4.1 Componentes de Portal . . . 85
4.4.2 Componentes de Banco de Dados . . . 87
4.4.3 Componentes de Grade . . . 88
4.5 Testes Aplicados . . . 91
4.6 Avaliação . . . 94
4.7 Conclusão . . . 96

5 Conclusões **98**

Referências Bibliográficas **100**

A Administração e gerenciamento do IAGP **106**

B Padrão JSR-168 **110**

C Descritores de Segurança **116**
C.1 Descritores de Contêiner . . . 117
C.2 Descritores de Serviço . . . 118
C.3 Descritores de Cliente . . . 118
C.4 Configurando o servidor . . . 119

Lista de Figuras

2.1 Conceito de criptografia simétrica, adaptado de Sotomayor (2005) 8
2.2 Conceito de criptografia assimétrica, adaptado de Sotomayor (2005) 10
2.3 Estrutura de um certificado digital, adaptado de Burnett et al. (2001) 12
2.4 Controle de Acesso 20
2.5 Controle de Acesso DAC, adaptado de Russell e Gangemi (1991) 21
2.6 Controle de Acesso MAC, adaptado de Russell e Gangemi (1991) 24
2.7 Controle de Acesso RBAC, adaptado de Russell e Gangemi (1991) 26

3.1 Diagrama em camadas GT4, OGSA, WSRF e serviços *web*, adaptado de Sotomayor e Childers (2006) 32
3.2 Relacionamento entre GT4, OGSA, WSRF e serviços *web*, adaptado de Alliance (2007b) 33
3.3 Arquitetura do GT4 (Jacob et al., 2005) 34
3.4 Comparação entre certificados X.509 e *proxy* 39
3.5 Processo de autenticação única, adaptado de Welch et al. (2004) 40
3.6 Delegação de um certificado *proxy* sobre um canal seguro, adaptado de Welch et al. (2004) 40
3.7 Visão geral do MyProxy, adaptado de Basney et al. (2005) 42
3.8 Funcionamento do CAS, adaptado de Chakrabarti (2007) 45
3.9 Arquitetura de autorização do GT4, adaptado de D.W.Chadwick et al. (2006) . . 46
3.10 Sistema de autorização VOMS, adaptado de Alfieri et al. (2005) 47
3.11 Infra-estrutura de autorização PERMIS, adaptado de Chadwick e Otenko (2003) 49
3.12 Modelo de autorização do AKENTI, adaptado de Thompson et al. (2003) 50
3.13 Diferentes funções do GSI, adaptado de Welch (2005) 52
3.14 Construção de páginas do Portal, adaptado de Klaene (2004) 54
3.15 Arquitetura de um *portlet* baseado em portal de grade, adaptado de Cai et al. (2006) 55
3.16 Grid Portlets tem uma arquitetura em camadas, adaptado de Novotny et al. (2006) 57
3.17 Arquitetura do OGCE, adaptado de Zhang et al. (2007) 60
3.18 Arquitetura do Clarens, adaptado de Lingen et al. (2005) 62

4.1 Funcionamento básico do IAGP 73
4.2 Visão geral da proposta IAGP 77
4.3 Casos de uso dos atores do IAGP 79
4.4 Autenticação do IAGP 81
4.5 Autorização do IAGP 83
4.6 Modelo de Dados pelo componente IAGP-AuthzRules 87
4.7 Fragmento do arquivo WSDD 89
4.8 Descritor de Segurança 89
4.9 Modelo de Autorização do GT4 90

A.1 Nova interface para autenticação dos usuários oferecida pelo IAGP 106
A.2 Lista de papéis registrados no portal. 107

A.3 Lista de usuários cadastrados no portal GridSphere 107
A.4 Definição de papéis para o usuário no GridSphere 108
A.5 Serviços a serem protegidos pelo IAGP 108
A.6 Associação do papel ao serviço pelo IAGP 109
A.7 Relatório integrado de atividades dos usuários do IAGP 109

B.1 Ciclo de vida de um *portlet* 111
B.2 Interação do usuário com um portal, adaptado de Microsystems (2003) 114

C.1 Descritor de segurança de contêiner (Alliance, 2006) 117
C.2 Configuração do descriptor do lado servidor (Alliance, 2006) 118
C.3 Descritor de segurança de serviço (Alliance, 2006) 118
C.4 Descritor de segurança do cliente (Alliance, 2006) 119

Lista de Tabelas

4.1 Papéis permitidos aos usuários. 92
4.2 Papéis requeridos para acesso aos serviços. 93
4.3 Classes de Equivalência para aplicação IAGP. 93
4.4 Casos de Testes para todas as classes. 93

Lista de Abreviaturas e Siglas

3-DES Triple Data Encryption Standard

AC Autoridade Certificadora

ACL Access Control List

AES Advanced Encryption Standard

ANS.1 Abstract Syntax Notation One

AR Autoridade de Registro

AOS Arquitetura Orientada a Serviço

API Application Programming Interface

CAS Community Authorization Service Architecture

CERN Conseil Européen pour la Recherche Nucléaire

CMS Compact Muon Solenoid Technical Proposal

CoG Kit Commodity Grid Kit

DAC Discretionary Access Control

DAI Data Access and Integration

DES Data Encryption Standard

DMTF Distributed Management Task Force

DN Distinguished Names

DRS Data Replication Service

EGA Enterprise Grid Alliance

EAD Ensino à Distância

FQHN Fully-Qualified Host Name

FTP File Transfer Protocol

GGF Global Grid Forum

GRAM Grid Resources Allocation Manager

GSI Globus Security Infrastructure

GSS Generic Security Services

GT Globus Toolkit

GTCP Grid TeleControl Protocol

HTML HiperText Markup Language

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol over Secure Socket Layer

IEC Internacional Electrotechnical Commission

IETF Internet Engineering Task Force

ISO Internacional Organization for Standardization

ITU-T Internacional Telecommunication Union

J2EE Java two Enterprise Edition

JAAS Java Authentication and Authorization Service

JCP Java Community Process

JDBC Java Database Connectivity

JSR Java Specification Request

JSP Java Server Pages

LDAP Lightweight Directory Access Protocol

LSF Load Sharing Facility

MAC Mandatory Access Control

MD-5 Message Digest Algorithm 5

MDS Monitoring and Discovery System

NIST National Institute of Standards and Technology

OASIS Organization for the Advancement of Structured Information Standards

OGF Open Grid Forum

OGSA Open Grid Services Architecture

OGSI Open Grid Services Infrastructure

OV Organização Virtual

PAM Pluggable Authentication Modules

PBS Portable Batch System

PDP Policy Decision Point

PDA Personal Digital Assistant

PEP Policy Enforcement Point

PGP Pretty Good Privacy

PIP Policy Information Point

PKI Public Key Infrastructure

RBAC Role-based Access Control

RC2 Ron's Code 2

RC4 Ron's Code 4

RFT Reliable File Transfer

RLS Replica Location Service

SAML Security Assertion Markup Language

SGE Sun Grid Engine

SHA-1 Secure Hash Algorithm

SOAP Simple Object Access Protocol

SSL Secure Socket Layer

SSO Single Sign On

TDB Trivial DataBase

TLS Transport Layer Security

URI Universal Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

WML Wireless Markup Language

WMS Workspace Management Service

WS Web Services

WSDL Web Services Description Language

WS-I Web Services Interoperability Organization

WSRP Web Services for Remote Portlets

XHTML eXtensible HyperText Markup Language

XML eXtensible Markup Language

XSLT eXtensible Stylesheet Language Transformations

Capítulo 1

Introdução

Grades computacionais (*Grid*) têm sido amplamente utilizadas em diversas áreas da ciência e engenharia como ferramenta de trabalho para cientistas, pesquisadores, engenheiros e técnicos que trabalham de forma colaborativa (Foster e Kesselman, 2004). Tendo como aliados a Internet e computadores de alto desempenho, as grades estão mudando a maneira de se fazer ciência. Diversos projetos fornecem exemplos que exploram intensamente o uso de grades computacionais em diversas áreas como a medicina, astronomia, química, engenharia civil, ecologia, geologia, física, entre outras (Alliance, 2007a).

Grades computacionais são plataformas para execução de aplicações que apresentam características diferentes das plataformas existentes. Segundo Baker et al. (2002), as principais características de uma grade computacional são:

- As grades agregam recursos geograficamente distribuídos em domínios administrativos distintos, conservando a autonomia administrativa de cada organização participante sobre os recursos que ela própria oferece.
- O ambiente de uma grade computacional tipicamente envolve uma variedade de recursos de natureza heterogênea e de uma ampla variedade de tecnologias.
- Uma grade pode ser composta de alguns poucos recursos até alguns milhões.
- Falhas dentro de um ambiente de grade que pode envolver grandes quantidades de recursos são bastante freqüentes e por isso gerenciadores de recursos devem se adaptar de modo a continuar operando e utilizando recursos de forma eficiente.

Grades computacionais são a combinação de uma infra-estrutura de rede e um arcabouço (*framework*) de software para disponibilizar recursos ou serviços baseados em recursos distribuídos de hardware e software. Grades permitem que comunidades (Organizações Virtuais) compartilhem recursos distribuídos geograficamente sem a necessidade de um controle ou gerenciamento centralizado. Elas normalmente oferecem para os seus usuários uma série de funcionalidades essenciais como por exemplo: autenticação de usuários, gerenciamento de transferência de arquivos, submissão de tarefas e monitoramento de recursos e serviços.

A segurança é uma questão importante em qualquer sistema computacional. A preocupação com segurança é particularmente importante e desafiadora para grades computacionais que têm recursos e usuários pertencentes a múltiplas instituições e domínios administrativos. Os componentes básicos de segurança de uma grade computacional devem implementar e disponibilizar mecanismos de autenticação, autorização, rastreabilidade (*accounting*) e uma comunicação segura entre remetente e destinatário. Os mecanismos de segurança de uma grade contribuem para o não comprometimento dos dados processados e sua exposição a terceiros e, também, para o uso correto dos recursos e serviços disponibilizados.

Para que se possa utilizar qualquer recurso em uma grade computacional, o usuário deve ter sua identidade verificada por meio de um processo de autenticação. Uma vez confirmada a identidade do usuário pelo processo de autenticação, a grade passa a avaliar quais são suas permissões antes de conceder acesso aos recursos solicitados. A verificação de autorização normalmente é feita com base em políticas de acesso aos recursos.

Para atender aos requisitos de segurança de uma grade computacional existem diversos mecanismos que são agregados ao ambiente: i) Segurança nos canais de comunicação por meio de criptografia e certificados digitais. ii) Autenticação mútua entre as partes comunicantes por meio de certificados e assinaturas digitais. iii) Políticas de controle de acesso aos recursos (autorizações) que está fortemente relacionada ao objetivo deste trabalho.

Na ferramenta Globus Toolkit (GT), uma das plataformas para computação em grade amplamente utilizada no mundo, o mecanismo de autenticação já está razoavelmente con-

solidado com o uso de certificados digitais X.509, um repositório de credenciais conhecido como MyProxy e pelo uso de um canal de comunicação segura que pode ser estabelecido por meio do protocolo TLS/HTTPS. Atualmente, diversos projetos são conduzidos em busca de soluções para autorização. Entretanto, as soluções para autorização em ambiente de grade ainda não estão devidamente consolidadas. Como exemplos de projetos que têm como objetivo tratar de aspectos relacionados à autorização em grades, podemos citar: AKENTI (Thompson et al., 2003), o PERMIS (Chadwick e Otenko, 2003) e o VOMS (Alfieri et al., 2003). Todavia, essas soluções de autorização são utilizadas em condições limitadas ou específicas e o assunto ainda é um desafio a pesquisa.

Uma alternativa interessante de acesso a recursos ou serviços de uma grade computacional por meio de uma interface *web* são os portais. Portais de grade (*Grid Portals*) permitem simplificar o acesso aos serviços de uma grade computacional, fornecendo um único ponto de acesso por meio de uma interface *web* padronizada. Portais de grade funcionam como uma ponte para um ambiente no qual o usuário pode fazer uso dos recursos da grade, tais como: execução de tarefas locais e remotas, recuperação e armazenamento de dados e acesso a recursos pertencentes a outros domínios e ainda trabalhar colaborativamente com outros usuários.

Com a crescente utilização dos portais de grade a questão da segurança ganha amplitude devido à natureza distribuída e multi-institucional. Nesse cenário com recursos pertinentes a vários domínios administrativos é preciso atender às boas práticas para segurança em redes de computadores (NIST, 2002). Deve haver mecanismos de identificação que garantam uma identidade única e verificável para cada usuário. Uma autoridade apropriada e confiável deve emitir a identidade do usuário. O sistema deverá possuir algum mecanismo de autorização que garanta que somente um usuário identificado possa obter acesso para somente o que está definido ao seu perfil pelo administrador do sistema. É necessário, também, um mecanismo para armazenamento das atividades dos usuários no sistema para fins de auditoria.

Da mesma forma que os portais, as grades computacionais devem implementar seus próprios mecanismos de segurança *in loco*. Por exemplo, o simples fato de um atacante obter privilégios indevidos que permita implantar código no portal e assim submeter tare-

fas à grade, não deve possibilitar que o mesmo invoque serviços, execute tarefas ou tenha acesso aos recursos da grade (Vecchio et al., 2006). Entretanto, a duplicidade de mecanismos de autenticação, autorização e rastreabilidade proporcionada pelas atuais soluções de portais de grades aliada à complexidade da arquitetura de muitas dessas aplicações favorece a ocorrência de erros de configuração a ponto de comprometer a segurança do sistema.

Há estudos sobre a segurança de portais que apontam para limitações e deficiências nas atuais soluções de portais de grades como, por exemplo:

- A questão de autorização é tratada de forma separada pelo portal e pela camada que regula o acesso aos recursos da grade. Por exemplo, portais criam seus próprios mecanismos de controle de acesso na interface com o usuário e que são distintos daqueles mecanismos que controlam o acesso *in loco* (junto ao recurso) gerando dificuldades e inconsistências. Nesse sentido é preciso criar ferramentas de administração integradas, que permitam administrar as permissões para a camada de acesso aos recursos e o portal conjuntamente.

- As soluções atuais muitas vezes são soluções de compromisso. Por exemplo, mapear todas as identidades do portal para uma única conta no *grid* pode oferecer simplicidade, mas não permite o estabelecimento de diferentes níveis de acesso para diferentes usuários. Dessa forma, é difícil determinar *a posteriori* "quem" fez "o quê", pois vários usuários compartilham a mesma identidade.

- O crescimento de aplicações para portais de grades computacionais exige dos desenvolvedores conhecimento sobre os detalhes dos mecanismos de segurança de grades que são difíceis de serem assimilados e integrados à aplicação. Atualmente, há uma carência de ferramentas que facilitem o desenvolvimento de aplicações para portais e permitam proteger os recursos tanto no lado do cliente como no lado do provedor de serviços.

- A má integração nos registros das atividades dos usuários, dificulta o trabalho dos administradores na correlação de eventos entre o portal e a grade. Por exemplo, no caso de registros separados, seria difícil para um administrador atender a uma

dificuldade operacional de algum usuário, sem envolver um exaustivo esforço de correlação de registros entre o portal e a grade.

1.1 Objetivos

O objetivo da presente pesquisa é propor um arcabouço de autorização para acesso a serviços de uma grade computacional por meio de uma solução de portal. A proposta é utilizar um modelo de autorização baseada em papéis para simplificar as regras de controle de acesso aos recursos de forma integrada com a solução de portal. Dessa forma, o gerenciamento de usuários e papéis poderá ser feito com as próprias ferramentas do portal. O mecanismo de autenticação da grade baseado em certificado X.509 pode ser integrado ao portal para oferecer um mecanismo de autenticação único. Além disso, um controle de atividades de usuários integrado facilita o processo de auditoria pelos administradores.

1.2 Metodologia

Para atingir os objetivos e propor uma solução para o problema foi analisada uma aplicação que provê suporte a produção colaborativa de objetos de aprendizagem em larga escala, por meio de uma grade computacional (Stanzani, 2008). A partir do estudo dessa aplicação foi levantado um conjunto de requisitos e necessidades com especial foco na autorização e controle de acesso. Com base nesse levantamento será elaborada uma proposta que atenda às necessidades de aplicações com perfil semelhante. Além disso, foi feita uma revisão da literatura recente sobre autorização em grades computacionais. Com base na revisão da literatura e nos levantamentos de requisitos será elaborada uma proposta de uma arquitetura com a implementação de um protótipo para prova de conceito.

1.3 Organização da Dissertação

No Capítulo 2 serão apresentados os fundamentos de segurança de sistemas computacionais. No Capítulo 3 serão apresentados os principais conceitos relacionados ao ambiente

de grades computacionais, a ferramenta Globus Toolkit, as principais soluções de portais de grade. No Capítulo 4 será apresentada uma proposta de uma arquitetura para integração e consistência das políticas de autorização para uma solução de portais de grade. E finalmente, o Capítulo 5 apresenta as conclusões.

Capítulo 2

Segurança de Sistemas

Este capítulo apresenta uma revisão bibliográfica sobre os principais fundamentos tecnológicos de segurança de sistemas computacionais, que são importantes no entendimento dos próximos capítulos.

2.1 Criptografia

As técnicas criptográficas permitem que o remetente de uma mensagem codifique os dados de modo que terceiros não possam compreender a informação, caso tenham conseguido, de alguma forma, interceptá-la. Somente o destinatário saberá restaurar a mensagem à sua forma original e compreender seu significado (Kurose e Ross, 2006). Há duas principais classes de algoritmos criptográficos: simétrica e assimétrica.

Os algoritmos de criptografia simétrica são baseados no uso de uma única chave para executar a operação de cifrar e decifrar os dados, conforme ilustra a Figura 2.1. Para assegurar que as mensagens trocadas entre o remetente e o destinatário sejam reconhecidas somente pelos mesmos, a chave deve ser distribuída de uma forma segura. Se alguém mais obtiver acesso a essa chave, que é usada para cifrar os dados, este poderá ser capaz de decifrar a informação contida na mensagem (Jacob et al., 2005).

As chaves são um componente importante dos algoritmos criptográficos simétricos e assimétricos. A chave criptográfica tem a utilidade semelhante à chave de uma fechadura de porta que permite a passagem pela porta por somente àqueles que a possuem. Cada

fechadura possui uma chave específica para abri-la. A chave deve ter certo tamanho e uma quantidade de ranhuras que permita o encaixe da chave na posição correta. A chave de um determinado fabricante de fechadura irá encaixar em qualquer fechadura de um determinado modelo, mas somente uma será a chave correta que permitirá abri-la. As chaves criptográficas são similares às chaves de uma fechadura de porta em muitos aspectos. Cada algoritmo criptográfico necessita de uma chave de valor e tamanho correto. Algoritmos simétricos utilizam a mesma chave para operação de cifrar e decifrar as mensagens. Pode-se executar o algoritmo com qualquer chave que tenha o tamanho correto, mas somente a chave que tenha o valor correto irá permitir a decodificação da mensagem cifrada. Por exemplo, chaves simétricas são chaves de mesmo tamanho e de mesmo valor, escolhida aleatoriamente. Se um algoritmo simétrico é de 40 bits significa que a chave para cifrar e decifrar tem o tamanho de 40 bits (Nash et al., 2001).

Figura 2.1: Conceito de criptografia simétrica, adaptado de Sotomayor (2005)
.

Alguns algoritmos mais comuns de criptografia simétricas são:

- *Data Encryption Standard* (DES): Esse algoritmo foi desenvolvido pela IBM em meados da década de 70. DES codifica o texto aberto em porções de 64 bits usando uma chave de tamanho fixo de 56 bits. DES foi muito usado como solução criptográfica por várias organizações e, principalmente, pela área financeira, mas devido ao crescimento do poder computacional uma chave de 56 bits tornou-se vulnerável a ataques de força bruta pela primeira vez em 1977 (Nash et al., 2001).

- 3-DES: Esse é um algoritmo que utiliza a combinação de três operações de DES e trabalha com chaves de tamanho fixo de 168 bits. Teoricamente mais seguro, mas os especialistas em análise de algoritmos criptográficos já conseguiram descobrir uma

vulnerabilidade no algoritmo 3-DES, reduzindo um ataque de força bruta de uma chave de 3-DES 168 bits para um equivalente de 108 bits (Nash et al., 2001). Outro problema do 3-DES é a necessidade de um esforço três vezes maior para cifrar uma mensagem.

- *Rons Code* 2 (RC2) e *Rons Code* 4 (RC4): Eles são algoritmos criptográficos de chave de tamanho variável, geralmente entre 40 a 128 bits. Fazem parte da série de algoritmos RC desenvolvido por Ron Rivest da RSA (Nash et al., 2001).

- *Advanced Encryption Standard* (AES): Esse é um algoritmo que processa dados em blocos de 128 bits e pode trabalhar com chaves de comprimento de 128, 192 e 256 bits. O AES é baseado no algoritmo chamado de Rijndael, desenvolvido por dois pesquisadores belgas. Rijndael foi o algoritmo vencedor de uma competição organizada pelo *National Institute of Standards and Technology* (NIST) para definir um algoritmo substituto do DES (Burnett et al., 2001).

Por mais de 2000 anos, desde a época de Julio César até o final dos anos 70, a comunicação cifrada exigia que duas partes comunicantes compartilhassem um segredo comum, uma chave que seria usada para cifrar e decifrar as mensagens. Em 1976, Whitfield Diffie e Martin Hellman, apresentaram o primeiro conceito de algoritmo de criptografia assimétrica ou de chave pública. O algoritmo de Diffie-Hellman é baseado em matemática de logarítmico discreto, cuja dificuldade de computação é igual ou superior à fatoração do produto de números primos. Atualmente, o algoritmo de chave pública de maior sucesso é o RSA, cujo nome se deve aos seus inventores R. Rivest, A. Shamir e L. Adleman. O RSA foi proposto em 1978 e se tornou um padrão nessa área, a ponto de criptografia de chave pública e RSA serem freqüentemente usados como sinônimos (Kurose e Ross, 2006).

Os algoritmos assimétricos utilizam duas chaves, sendo uma pública e a outra privada. A chave usada para cifrar a mensagem é diferente daquela usada para decodificá-la. A chave privada deve ser protegida do acesso de terceiros, enquanto que a chave pública pode ser conhecida por todos. Os algoritmos de chave pública atestam que as chaves pública e privada devem ser geradas de forma que uma mensagem cifrada por uma delas só poderá ser decifrada com a outra chave que pertence ao mesmo par. Mesmo de posse de uma

delas, é computacionalmente inviável encontrar seu par. Por exemplo, uma mensagem cifrada com a chave pública só pode ser decodificada com a chave privada correspondente a esse mesmo par, conforme ilustra a Figura 2.2.

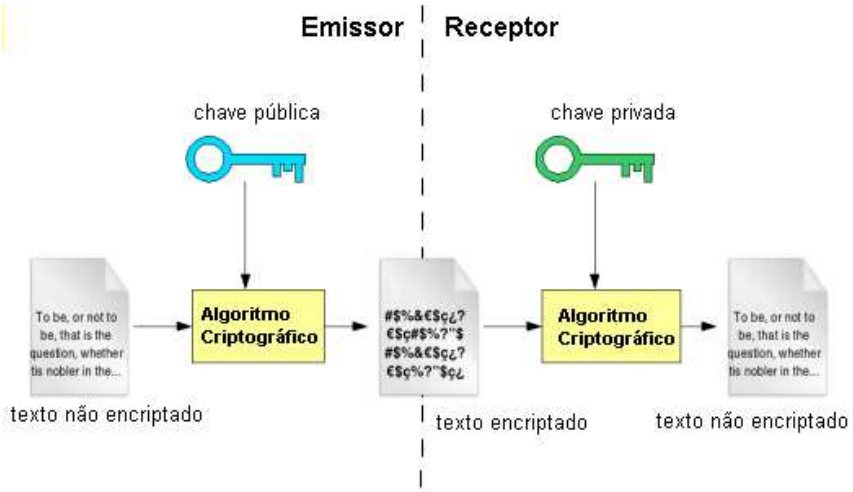

Figura 2.2: Conceito de criptografia assimétrica, adaptado de Sotomayor (2005)

2.2 Assinatura Digital

Assinatura digital é uma técnica baseada em métodos criptográficos de chave pública. Essa técnica tem resultado análogo ao processo de assinaturas escritas a mão, onde uma entidade pode ler a assinatura digital e verificar sua autenticidade. Todavia, essa técnica é muito segura porque é computacionalmente impraticável para qualquer outra entidade forjar uma assinatura digital (Adams e Lloyd, 2002). Assinaturas digitais devem ser passíveis de verificação, não admitindo adulteração, ou mesmo, o repúdio do assinante da mensagem.

2.3 Certificados Digitais

Um certificado digital é um documento digital que garante que uma chave pública pertence a uma determinada entidade. Ele possui uma série de campos, como por exemplo, a identificação do proprietário do certificado e a sua chave pública. A partir de uma função *hash*, gerada com o conteúdo do certificado do usuário, a Autoridade Certificadora (AC) assina digitalmente o certificado, cifrando-o com o *hash* de sua chave privada. Uma

Autoridade Certificadora é a entidade responsável pela assinatura e emissão dos certificados. Assim, a Autoridade Certificadora garante que somente sua chave pública pode ser usada para ler o certificado e portanto, reconhecê-lo como válido. Para operação de validação do certificado é necessário somente conhecer a chave pública da Autoridade Certificadora, que verifica a validade da assinatura da Autoridade Certificadora que consta no certificado. Se a verificação for positiva, então o certificado está íntegro e a Autoridade Certificadora reconhece a chave pública do usuário que consta no mesmo. Portanto, a mensagem assinada pelo usuário, pode ser decifrada com a chave pública que consta no certificado. A chave pública da Autoridade Certificadora deve ser previamente conhecida pela aplicação em questão (por exemplo, deve estar armazenada no aplicativo de acesso à Internet).

Certificados digitais são construídos utilizando formatos padronizados, como por exemplo certificado de chave pública X.509 (Housley et al., 2002). Além desse, existem outros formatos como por exemplo: certificado *Pretty Good Privacy* (PGP) e certificado de atributos (Infra-estrutura de gerenciamento de privilégio) (Burnett et al., 2001). O formato mais amplamente utilizado atualmente é o padrão X.509 versão 3 da ITU-T.

O padrão ITU-T X.509 foi inicialmente proposto e publicado em 1988 pela *International Telecommunication Union - Telecomunication* (ITU-T), tornando-se conhecido como o padrão ITU-T X.509 ou ISO/IEC 9594-8. Ele é parte das recomendações do padrão de Serviço de Diretório X.500 (ITU-T, 2000). Em 1993, devido à revisão do X.500, foram adicionados mais alguns melhoramentos, o que resultou no lançamento da versão 2. Entretanto, os certificados baseados nas versões 1 e 2 eram muito deficientes em vários aspectos, principalmente em relação à falta de alguns campos. Então, em junho de 1996, a versão 3 foi publicada. Entretanto, o padrão ISO/IEC/ITU e ANSI X9 é muito amplo para sua aplicabilidade na Internet. Em razão disso, foi definido um grupo de trabalho chamado *Public Key Infrastructure* X.509 (PKIX) do *Internet Engineering Task Force* (IETF) para criar um padrão mais adequado de certificados X.509 que atendesse às necessidades da Internet (Housley et al., 2002). Em 1999, o IETF definiu esse padrão, RFC2459 baseada na versão ITU versão 3 (Housley et al., 1999). Em abril de 2002, a RFC2459 foi substituída pela RFC3280 (Housley et al., 2002).

Quando há necessidade de transferir dados cifrados para outra entidade, torna-se necessário que os dados obedeçam a um padrão de formato com sintaxe e codificação apropriadas para que haja interoperabilidade entre as aplicações e usuários. Portanto, existem regras e uma sintaxe para codificação dentro do padrão X.509, que são expressas utilizando uma notação especial conhecida como *Abstract Syntax Notation 1* (ASN.1) que foi originalmente criada pela *Open Systems Interconnection* (OSI) para a série de protocolos X.500 (Burnett et al., 2001).

Um certificado de chave pública X.509 é geralmente um arquivo texto que contém um identificador único baseado na convenção de nomes do padrão X.500 conhecido como DN (*Distinguished Name*) e alguns outros campos que contém informações sobre o proprietário (usuário ou recurso) e sua chave pública. O exemplo a seguir ilustra o identificador do proprietário de um certificado:
DN:/O=Grid/O=InterGrid/OU=unisantos.edu.br/CN=Aluno.

A Figura 2.3 ilustra melhor todos esses campos nas diferentes versões de certificados que a seguir estão descritos:

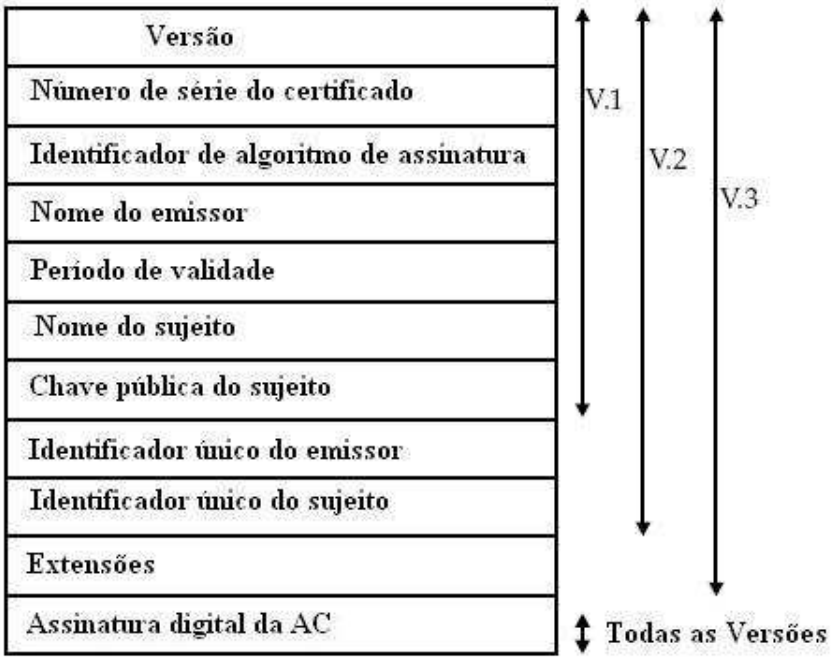

Figura 2.3: Estrutura de um certificado digital, adaptado de Burnett et al. (2001)

1. Versão: campo que diferencia as versões atuais e futuras. Exemplo: V3

2. Número de série do certificado: é um campo que contém um valor inteiro e único, gerado pela AC. Exemplo: 59 83 20 9b ce d5 1e d0 9a 5e b6 50 68 49 0c 2f

3. Identificador de algoritmo de assinatura: esse campo contém um identificador do algoritmo usado para calcular a assinatura junto com outros parâmetros associados. Exemplo: sha1RSA

4. Nome do emissor: este campo identifica quem emitiu e assinou o certificado, através de um *Distinguished Name* ou simplesmente DN (Howes et al., 1995). Exemplo: DN:/E=premium-server@thawte.com/CN=Thawte Premium/C=ZA

5. Período de validade: esse campo tem duas datas que define o início e o fim do período de validade do certificado. Exemplo: quarta-feira, 7 de março de 2007 18:06:28/sexta-feira, 7 de março de 2008 18:06:28

6. Nome do sujeito: este campo contém o proprietário associado a esse certificado. Exemplo:
   DN:/CN=www2.bancobrasil.com.br/OU=UGS/O=Banco do Brasil S.A./C=BR

7. Chave pública do sujeito: esse campo contém a chave pública do proprietário desse certificado. Exemplo de uma chave pública de um certificado do site do Banco do Brasil: c6 37 68 a0 32 e8 0e 4f 50 83 b8 a7 e2 c6 26 c9 5c 3e 26 7b de 23 2c 6a 4a 6c 08 3f a8 37 48 5a b2 1e 9b 71 00 f4 4b 7d 60 02 2e 48 44 74 e5 dc d0 a6 d2 8d 67 6d f6 22 97 38 74 57 ee bc d7 44 1a b3 01 6b da 90 26 4e 1e 98 56 0c 9a f6 5c b8 42 68 42 55 61 96 a0 8e 1c be e6 a5 e9 75 ca 83 c8 f3 06 b1 4c c9 33 9e 45 29 e1 a9 9d bf 4f a5 11 92 f1 0c 75 bc 11 52 73 8a f0 57 16 4f 3b 45

8. Identificador único do emissor: é um identificador único para certificados na versão 2 e 3. Esse campo raramente é usado na prática (reutilização de nomes) e não é recomendado pela RFC3280 (Housley et al., 2002).

9. Identificador único do sujeito: é um identificador único para certificados na versão 2 e 3. Esse campo raramente é usado na prática (reutilização de nomes) e não é recomendado pela RFC3280.

10. Extensões: estão presentes somente na versão 3. Essas extensões compreendem informação sobre política e chave, atributos adicionais do sujeito e emissor do certificado e restrições de uso.

11. Assinatura digital da AC: esse campo contém a assinatura da entidade emissora do certificado (Autoridade Certificadora). Exemplo: 4c 82 48 00 59 98 b8 df c4 0e c5 69 2a 11 83 db bb d0 3c 13.

2.4 Autoridade Certificadora

A vinculação de uma chave pública a uma entidade em particular é feita, tipicamente, por uma Autoridade Certificadora (AC), cuja tarefa é validar identidades e emitir certificados digitais (Kurose e Ross, 2006). O registro de uma entidade junto a uma AC pode ser obtido após algumas etapas. Primeiramente, a entidade fornece uma prova de identidade para a AC. Após uma identificação positiva, um certificado é gerado e assinado digitalmente pela AC, o que significa que ela reconhece a identidade dessa entidade. O certificado contém a chave pública da entidade e informações que identificam quem é o proprietário do certificado. As principais responsabilidades de uma AC são (Jacob et al., 2005):

- Obter uma identificação positiva das entidades requisitantes de certificados.
- Publicar, remover e arquivar os certificados.
- Garantir proteção aos certificados e ao servidor onde está instalada a AC.
- Manter nomes únicos para os proprietários dos certificados.
- Fornecer certificados assinados para entidades que precisam ter sua identidade verificada.
- Manter atividades de registro de acesso.

Antes que uma AC possa assinar certificados digitais de seus clientes, ela deve executar o mesmo procedimento para assinar o seu próprio certificado. Nesse ponto está

sendo considerada uma infra-estrutura com uma única AC que é executada em alguns passos. Primeiramente, a AC gera seu próprio par de chaves: pública e privada. Depois, armazena em local seguro sua chave privada e cria seu próprio certificado. Finalmente, assina seu certificado com sua chave privada. Um dos pontos mais importantes da segurança de uma infra-estrutura de chave pública é proteger a chave da privada da AC de terceiros. Entretanto, para garantir a segurança de uma AC é necessário atender a outros itens de segurança, como restrições físicas e remotas, monitoramento e auditagem do servidor (Jacob et al., 2005).

Em alguns ambientes, uma Autoridade de Registro (AR) também pode fazer parte dessa infra-estrutura. Uma AR atua como intermediária entre a AC e as entidades (clientes), com o objetivo de diminuir a carga de serviço da AC. A AR é responsável por aprovar e rejeitar as requisições de certificados e repassar as informações sobre o usuário para a AC. Ela, também verifica e garante que a informação do cliente esteja correta antes que o certificado seja emitido. As ARs são especialmente importantes para a escalabilidade de uma Infra-estrutura de Chave Pública (ICP) que permite interligar diferentes localizações geográficas. Por exemplo, uma AC pode delegar responsabilidades para diferentes ARs e designar uma área de operação para cada AR que pertence ao seu domínio de atuação (Choudhury et al., 2002).

Existem algumas formas de estabelecer relações de confiança entre ACs. Uma alternativa é um modelo hierárquico e a outra é o ponto a ponto. O modelo ponto a ponto assume o estabelecimento de uma relação de confiança entre duas ACs que não são consideradas subordinadas uma da outra, normalmente estão em domínios administrativos diferentes. Esse modelo é mais conhecido como certificação cruzada (*cross certification*), porém tem a desvantagem de não ser escalável para um número elevado de relações (Nash et al., 2001).

O modelo hierárquico é uma solução mais adequada ao ambiente corporativo ou empresarial. À medida que o número de usuários cresce, torna-se difícil para a AC fazer a verificação dos certificados de todos seus clientes. Portanto, uma única AC passa a se tornar o gargalo do processo. Nesse caso, a solução mais adequada é utilizar uma hierarquia de certificado (*certificate hierarchy*) na qual uma AC delega sua autoridade para uma

ou mais autoridades subordinadas (Burnett et al., 2001). Essas autoridades, em cascata, podem ter suas próprias autoridades subordinadas. Um processo de busca para encontrar a AC que assinou o certificado navega dentro dessa estrutura de árvore invertida.

2.5 Autenticação

Autenticação é um processo em que uma entidade oferece provas sobre sua própria identidade à outra entidade. Em um ambiente de rede, quando duas entidades precisam se comunicar, utilizam um protocolo de autenticação que passa obrigatoriamente pela troca de mensagens e dados entre elas. O protocolo de autenticação deve estabelecer a identidade entre as partes de maneira satisfatória para ambas.

A segurança do processo de autenticação é geralmente relacionada com o número de fatores ou provas de identidade que são apresentadas durante o processo de estabelecimento de identidade. Um processo de autenticação pode ser feito de várias formas, mas a mais conhecida é a identificação por nome e senha. Ela é considerada um esquema de um único fator, pois é necessário somente que o usuário mostre o conhecimento da senha. Já um processo de autenticação de dois fatores requer que o usuário apresente, além do conhecimento da senha, a posse de algum dispositivo físico. Por exemplo, SecureID, da RSA Security, é considerado um mecanismo de autenticação de dois fatores, pois utiliza uma senha e um dispositivo de hardware para o processo de identificação do usuário (Nash et al., 2001).

2.6 Autorização

Autorização é o processo que verifica quais direitos ou permissões devem ser concedidos a uma entidade para acessar os recursos do sistema. O objetivo da autorização, também chamada de controle de acesso, é restringir o acesso aos recursos do sistema somente àquelas entidades que realmente necessitem dos mesmos. Além disso, o controle de acesso visa impedir o uso não autorizado de recursos, o acesso não autorizado de entidades, e limitar as ações de usuários autenticados pelo sistema (Vecchio et al., 2006).

Tipicamente se utiliza o princípio do privilégio mínimo, segundo o qual só deve ser concedida ao usuário uma autorização mínima necessária a executar as ações que lhe são pertinentes. A maneira mais comum de gerenciar autorizações é através de listas de controle de acesso (centralizado por usuário) que indica os sujeitos no sistema com acessos autorizados ao objeto considerado ou listas de competências (centralizada em recursos) que indica para cada objeto do sistema, quais as permissões de acesso cada sujeito possui (Vecchio et al., 2006).

Listas funcionam bem quando o conjunto de usuários e recursos são pequenos, mas é muito custoso gerenciar um conjunto muito grande de elementos. Uma forma de combater a complexidade é utilizar um controle de acesso baseado em papéis (Vecchio et al., 2006). Nesse modelo, os usuários são associados a papéis e estes regulam as atividades que eles executam no sistema. Os modelos de controle de acesso serão discutidos em maiores detalhes na seção 2.9.

2.7 Integridade

A integridade se refere à garantia de que a mensagem recebida não foi alterada ou corrompida durante a transmissão da informação. Existe uma classe de algoritmos que são usadas para prover integridade às mensagens. Eles são conhecidos como resumos de mensagens (*Message Digests*). Esses algoritmos utilizam uma função *hash* que recebe uma mensagem como entrada e retorna uma mensagem de tamanho fixo, de forma que seja inviável a conversão para a mensagem original. Vale observar que a quantidade de colisões (resultados iguais para mensagens diferentes) deve ser muito pequena. Qualquer alteração na mensagem pode ser detectada porque será gerada um resumo diferente pelo algoritmo (probabilidade muito alta que isso aconteça). Alguns dos algoritmos de resumos de mensagens mais conhecidos são: *Message-Digest Algorithm 5* (MD-5) e *Secure Hash Algorithm* (SHA-1).

O MD-5 é o algoritmo de resumo de mensagem desenvolvido por Ron Rivest e produz um resumo de mensagem de tamanho de 128 bits. O SHA-1 é outro algoritmo muito utilizado e adotado como padrão pelo governo americano no uso de aplicações oficiais

que produz um resumo de mensagem de tamanho de 160 bits. Um resultado com maior número de bits torna o algoritmo SHA-1 mais seguro (Kurose e Ross, 2006).

2.8 Comunicação Segura

A comunicação segura permite que duas partes que desejem trocar mensagens entre si, estabeleça um canal de comunicação baseado em três pilares: privacidade, integridade e autenticação. Alguns autores adicionam o não repúdio como um quarto pilar, mas a maioria deles considera o não repúdio como parte do processo de autenticação (Sotomayor, 2005). Privacidade é uma característica geralmente obtida através da utilização de algum algoritmo criptográfico onde somente o emissor e o destinatário são capazes de entender a mensagem. Integridade é a característica que assegura que a mensagem recebida pelo destinatário é exatamente aquela mensagem elaborada pelo emissor conforme comentado na seção 2.7. Autenticação assegura que as partes envolvidas no processo de comunicação são realmente quem diz ser conforme já explicado na seção 2.5.

A comunicação segura pode ser conseguida em níveis de: mensagem ou transporte. Apesar da proteção das mensagens prover maior segurança ao processo, a proteção em nível de transporte é, sem dúvida, a mais utilizada, como por exemplo, em sítios de comércio eletrônico na Internet. O protocolo mais usado em nível de transporte é o *Secure Socket Layer* (SSL). Ele originalmente foi desenvolvido pela Netscape Communications Corporation em meados dos anos 90. A versão 2 do SSL foi a primeira versão largamente utilizada na Internet. A versão 3 do SSL ofereceu mais flexibilidade, eficiência e um maior conjunto de facilidades (Nash et al., 2001). Em 1999, o IETF homologou o TLS (Transport Layer Security) (Dierks et al., 1999) versão 1 que englobou todas as funcionalidades da versão 3 do SSL (Housley et al., 1999). O SSL/TLS apresenta as seguintes características:

- Permite que um usuário confirme a identidade de um servidor *web*. O navegador *web* do cliente habilitado para SSL mantém uma lista de Autoridades Certificadoras de confiança (*Trust List*) em conjunto com as chaves públicas de cada uma delas. Quando um cliente acessa um sítio *web* que utiliza o protocolo SSL, ele recebe um certificado do servidor *web* que contém a chave pública do proprietário do mesmo em

um campo específico do certificado. O certificado foi assinado digitalmente por uma Autoridade Certificadora e que aparece nessa lista de Autoridades Certificadoras que o cliente confia. Assim, o navegador pode ter a identidade do servidor validada, antes de qualquer operação do cliente.

- Permite que o servidor confirme a identidade do cliente. Embora a autenticação do cliente seja suportada pelo protocolo SSL/TLS, sua utilização é opcional.

- Provê confidencialidade e integridade para as mensagens trocadas entre o servidor *web* e o cliente (sessão cifrada).

2.9 Modelos de Controle de Acesso

A finalidade do controle de acesso é limitar a ação ou operação que um usuário legítimo pode executar em um computador. O controle de acesso restringe o que um usuário diretamente pode fazer em um sistema computacional e define quais os programas podem ser executados em nome de usuários. Desse modo, o controle de acesso procura impedir atividades que poderiam levar a uma falha na segurança (Sandhu e Samarati, 1994).

O controle de acesso consiste de quatro elementos: sujeitos, objetos, operações e um monitor de referência. Os "sujeitos" são, na verdade, usuários e grupos de usuários. Objetos são arquivos e recursos como por exemplo: memória, impressoras, *scanner*, etc. Operações ou ações que podem ser feitos aos objetos pelos sujeitos originados de um acesso *web*, memória, servidor ou por meio de chamada de métodos (Rizza, 2005). O controle de acesso é forçado por meio de um monitor de referência que é o mediador de todas as requisições de acesso dos sujeitos aos objetos de um sistema. O monitor de referência consulta um banco de dados que contém um conjunto de regras que correspondem à política de segurança para determinar se um sujeito tem a autorização necessária para executar a operação que ele está pretendendo fazer (Sandhu e Samarati, 1994). A Figura 2.4 ilustra o controle de acesso.

O monitor de referência faz uso de regras de segurança através de controles, estabelecendo os limites de operação dos usuários no sistema e protegendo seus dados e recursos da ação de intrusos. A definição de políticas de segurança é normalmente orientada por

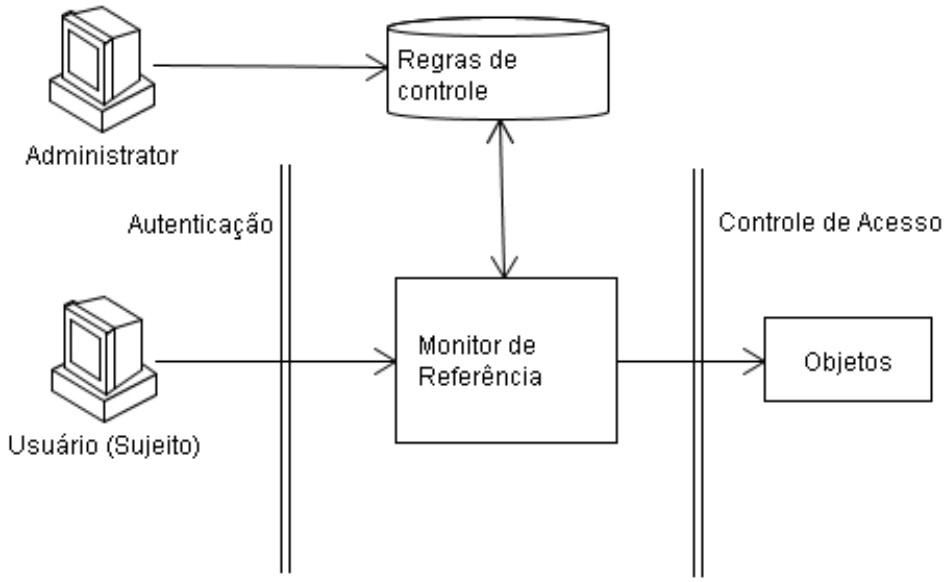

Figura 2.4: Controle de Acesso

modelos de controle de acesso que fornecem na representação abstrata o funcionamento seguro do uso no sistema (Lento et al., 2006). Os modelos de controle de acesso mais comuns na literatura são (Sandhu e Samarati, 1996): *Discretionary Access Control* (DAC), *Mandatory Access Control* (MAC) e *Role-Based Access Control* (RBAC).

2.9.1 Modelo DAC

Discretionary Access Control (DAC) é o modelo utilizado em boa parte dos sistemas atuais (Lento et al., 2006) onde a decisão da aplicação de regras associadas a uma política de segurança é deixada nas mãos do usuário final. É um mecanismo de controle de acesso que permite a um sujeito conceder permissão para que outros usuários possam acessar os seus dados. O exemplo mais comum é o sistema de arquivos de um computador onde o proprietário de um arquivo define se permite ou nega o acesso ao mesmo de acordo com a sua vontade. Esse mecanismo de controle de acesso é altamente flexível, pois oferece o melhor ponto de equilíbrio entre segurança e aplicabilidade. Entretanto, essa mesma característica o torna vulnerável a certos tipos de ataques como programas que executam ações maliciosas a partir de aplicações dos usuários (cavalos-de-tróia). É um modelo que não impõe nenhum tipo de controle sobre como a informação é propagada e usada, uma vez que ela tenha sido acessada por usuários autorizados (Rizza, 2005). A Figura 2.5 ilustra o controle de acesso DAC baseado em uma lista de controle de acesso onde Jane,

a proprietária do arquivo, permite que John tenha acesso a somente leitura ao arquivo de salário dos funcionários, mas não permite que Sam tenha acesso ao arquivo.

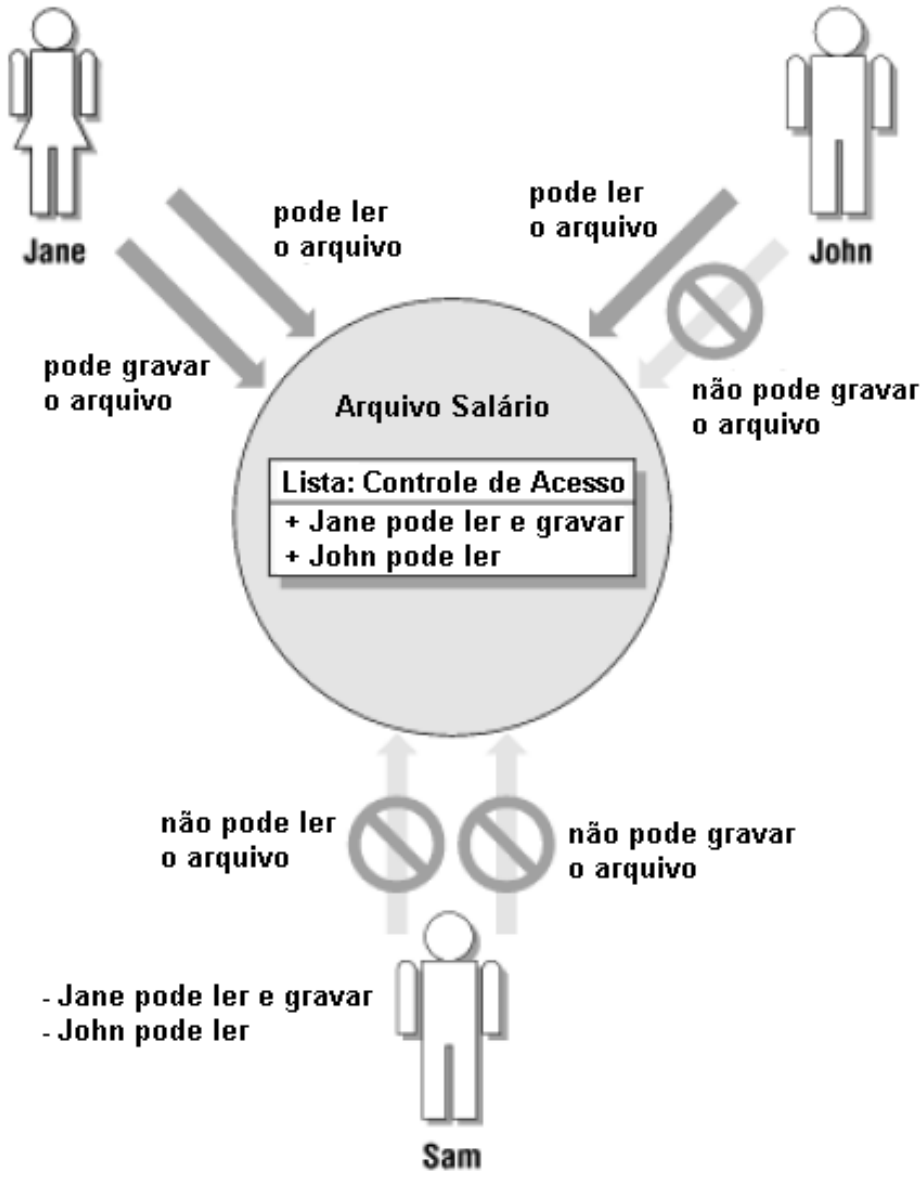

Figura 2.5: Controle de Acesso DAC, adaptado de Russell e Gangemi (1991)

2.9.2 Modelo MAC

Mandatory Access Control (MAC) é o resultado das pesquisas financiadas pelo Departamento de Defesa Americano na década de 70. O mecanismo de autorização no modelo MAC assegura um alto grau de proteção de maneira a impedir qualquer fluxo de informação não autorizado através da aplicação de regras de segurança em vários níveis. As políticas de segurança nesse modelo governam a classificação base dos sujeitos (por exem-

plos: usuários e aplicações) e dos objetos (por exemplo: arquivos, diretórios, dispositivos físicos, *sockets*, *windows*) dentro desse sistema. Cada objeto e cada usuário são associados a um nível de segurança, isto é, o potencial de risco que poderia resultar de um acesso não autorizado a informação. O nível de segurança associado com um usuário reflete a confiança no mesmo de não revelar uma informação importante para alguém não autorizado. Dentro de um exemplo muito simples, nível de segurança é um conjunto ordenado de elementos hierárquicos. Dentro da área militar e de algumas áreas governamentais, o conjunto hierárquico consiste de: Ultra-Secreto (US), Secreto (S), Confidencial (C) e Não Classificado (NC), onde US > S > C > NC. Dentro dessa estrutura hierárquica, cada nível de segurança é dominado por ele mesmo e por todos os outros abaixo dele. O acesso de um usuário a um objeto é concedido somente se alguma relação é satisfeita entre o nível de segurança desses dois conjuntos (Rizza, 2005). De uma maneira semelhante, uma Universidade poderia considerar outros níveis de segurança como por exemplo: aluno, professor e funcionário. Entretanto, a simples associação em níveis de segurança não garante um princípio básico da segurança ligado à quantidade de pessoas que tem acesso à informação. Segundo esse princípio, quanto menos pessoas compartilharem um segredo mais fácil será garantir que esse segredo não seja revelado. Para resolver esse problema foi criado o conceito de categorias que permite o acesso às diferentes categorias à medida que as suas incumbências demandem este acesso. Para implementar esses conceitos ao contexto computacional são definidos rótulos de segurança que agregam a informação dos níveis de segurança (classificação) e categorias (Russell e Gangemi, 1991). Cada sujeito e objeto dentro de um sistema MAC tem um rótulo de sensibilidade [1]associado com ele. O rótulo de sensibilidade consiste de duas partes: a classificação e as categorias. Por exemplo: no rótulo "SECRETO (VENUS, TANK, ALPHA)", "SECRETO" é sua classificação, e "(VENUS, TANK, ALPHA)" é sua categoria. O rótulo de um objeto define sua classificação (*classification*) e o rótulo do sujeito é chamado de autorização ou habilitação (*clearance*) do sujeito (Lento et al., 2006). A classificação é definida em um simples nível hierárquico, como demonstrado anteriormente em um ambiente militar (ULTRA SECRETO > SECRETO > CONFIDENCIAL > NÃO CLASSIFICADA).

[1]Segundo Lento et al. (2006) a tradução mais apropriada para *sensitivity* seria sensitividade, porém sensibilidade descreve melhor em Português, o aspecto semântico por trás dessas classificações. No texto desta dissertação, será adotada a mesma tradução.

As categorias são representadas de forma não hierárquica e representam áreas distintas de informação dentro de seu sistema. As categorias podem ser agrupadas em conjuntos como por exemplo, considerando o mesmo ambiente militar: SDI, TANK, SUB, VENUS, STEALTH. Em áreas comerciais, as categorias podem corresponder aos departamentos, nomes de produtos, anúncios de campanha, etc. A idéia é que mesmo que alguém tenha a mais alta classificação, a ele não seja confiado ter acesso a todos os níveis de informação (Russell e Gangemi, 1991). Esses esquemas são geralmente usados em ambientes que exigem altos níveis de segurança, onde a autorização e a classificação tornam-se um ponto muito importante para acesso aos objetos ou recursos (Chakrabarti, 2007). Por exemplo, para ver a informação da categoria VENUS, o indivíduo deve realmente ter a necessidade do conhecimento dessa informação. Existem regras para importar e exportar dados em sistemas baseado em MAC. Eles também, controlam quais dispositivos do sistema podem copiar ou imprimir a informação. Por exemplo: o usuário pode não ter permissão para imprimir informação sensível em impressoras localizadas em áreas públicas do edifício da sua empresa.

A Figura 2.6 ilustra o modelo de controle de acesso MAC. Em um sistema com controle de acesso MAC, todas as decisões são tomadas pelo sistema. A decisão de permitir ou negar acesso para um objeto (por exemplo, um arquivo de logística) envolve a interação de três elementos:

- O rótulo do sujeito (sua habilitação ou autorização). Por exemplo: "ULTRA SECRETO [VENUS TANK ALPHA]"
- O rótulo do objeto. Por exemplo, o rótulo de sensibilidade do arquivo de logística é "SECRETO [VENUS ALPHA]"
- A requisição de acesso. Por exemplo, a tentativa de ler o arquivo.

Quando John tenta ler o arquivo de logística, seu nível de sensibilidade deve ser igual ou superior ao nível SECRETO e a sua categoria deve incluir rótulos especificado no rótulo do arquivo. Nesse caso, John será capaz de ler o arquivo porque sua categoria inclui somente os rótulos VENUS e ALPHA, o mesmo exigido pelo arquivo.

Para que John possa alterar o conteúdo do arquivo, seu nível de sensibilidade deve ser igual ou menor que o nível de sensibilidade do arquivo e sua categoria deve incluir todos os rótulos que fazem parte do mesmo. Como John tem o rótulo ULTRA SECRETO e o arquivo tem o rótulo SECRETO, então, ele não pode alterá-lo. Isso evita que usuário com classificação elevada, ULTRA SECRETO, copie seções do documento e tente gravar em arquivo com classificação inferior, como por exemplo: NÃO CLASSIFICADO. Porém, esse modelo de controle de acesso permite elevar o nível de classificação da informação, por exemplo: de CONFIDENCIAL para SECRETO.

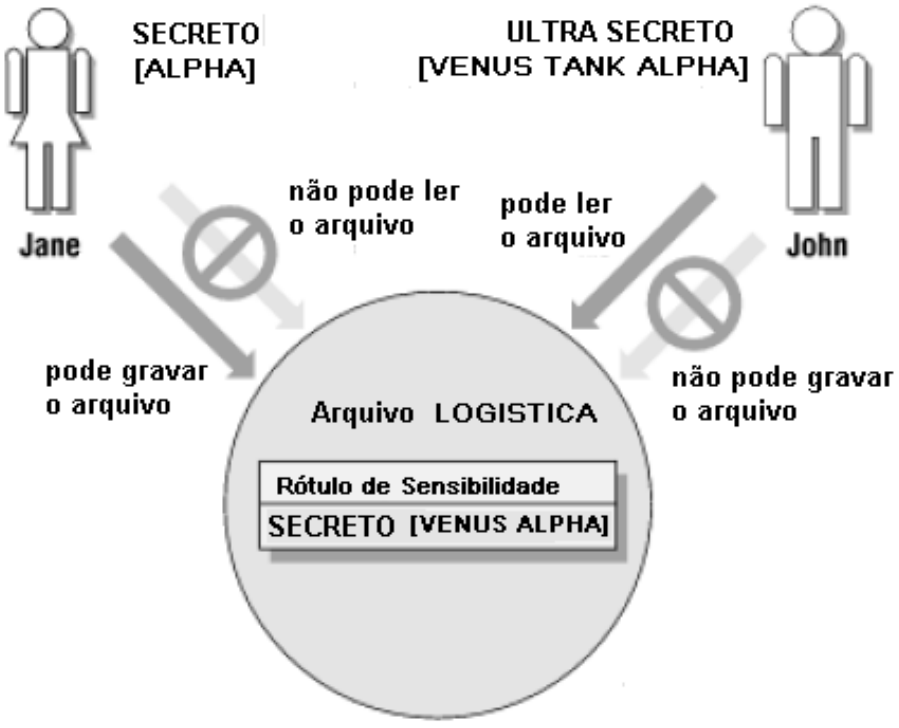

Figura 2.6: Controle de Acesso MAC, adaptado de Russell e Gangemi (1991)

2.9.3 Modelo RBAC

Role-Based Access Control (RBAC) é um modelo emergente que surgiu na década de 90 para prover segurança em sistemas de grande porte. Basicamente as permissões são associadas a papéis e usuários são mapeados para papéis. O modelo RBAC é independente de política. Diferente do que acontece com os modelos tradicionais de controle de acesso, o RBAC possibilita uma grande flexibilidade e facilidade do ajuste do controle de acesso à medida que ocorrem mudanças no ambiente (Lento et al., 2006). Esse modelo necessita da

identificação prévia dos papéis no sistema. O papel é definido como sendo um conjunto de atividades e responsabilidades associadas a um determinado cargo ou função. Com RBAC, os administradores de sistema podem criar papéis, definir permissões para esses papéis e então associar usuários para os papéis com base nas responsabilidades associadas a um determinada atividade. Por conseguinte, um usuário que está associado a um papel pode realizar todos os acessos para os quais o papel está autorizado. Nesse particular, a relação permissão-papel pode simplificar o mapeamento de usuários para papéis pré-definidos. Sem RBAC, seria mais difícil determinar quais permissões foram autorizadas para qual usuário (Park et al., 2001). RBAC é uma tecnologia mais recente que está tornando-se muito popular e tem atraído a atenção da comunidade devido ao seu potencial de reduzir a complexidade e o custo da administração da segurança em aplicações em grades redes.

O modelo RBAC possui diversas características (Sandhu e Samarati, 1994), tais como:

- Gerência de autorizações: a política baseada em papéis propicia uma lógica mais simples na especificação da autorização a qual pode ser dividida em duas partes: a associação de direitos de acesso aos papéis e associação de papéis a usuários. Isso reduz o custo de gerenciamento de permissões. Por exemplo, suponha que a responsabilidade de um usuário mudou devido a uma promoção. Com RBAC, o atual papel do usuário pode ser facilmente alterado para um novo papel que é associado às novas responsabilidades. Porém, se toda autorização fosse diretamente aplicada aos usuários e aos objetos, isso obrigaria a revogação de todos os direitos de acesso do usuário e uma nova associação deveria ser executada. O resultado dessa segunda proposta teria um elevado custo de gerenciamento.

- Suporte a hierarquias de papéis: em muitas aplicações existem algum tipo de hierarquia natural de papéis baseada em noções de generalização e especialização. Isso possibilita que permissões sejam herdadas e compartilhadas através da hierarquia. Por exemplo, os papéis de engenheiro de hardware e de software são especializações do papel engenheiro. Um usuário associado ao papel de engenheiro de hardware irá herdar os privilégios e permissões do papel engenheiro.

- Suporte a privilégio mínimo: os papéis permitem que um usuário trabalhe com o mínimo privilégio exigido para uma determinada tarefa. Usuários autorizados

a exercer papéis mais poderosos só precisam exercê-los quando for absolutamente necessário. Isso minimiza a possibilidade de danos por causa de erros inadvertidos ou por um intruso disfarçado de um usuário legítimo.

- Separação de tarefas: separação de tarefas se refere ao princípio que não deve ser dado a nenhum usuário, privilégio suficiente que possa permitir o uso indevido do sistema em seu próprio proveito. A separação de tarefas é obtida através de restrições à autorização e/ou ativação de papéis considerados mutuamente exclusivos (Lento et al., 2006). Por exemplo, uma pessoa autorizada a fazer o pagamento de salário não deve ser a mesma que libera o seu pagamento.

A Figura 2.7 ilustra o modelo RBAC, a usuária Jane está associada ao papel de PROFESSOR que permite acesso a qualquer objeto do sistema onde o papel de PROFESSOR tenha permissão de acessá-lo. Já John está mapeado com o papel de ALUNO que não tem permissão de acessar o objeto, cujo papel requerido está associado ao papel de PROFESSOR.

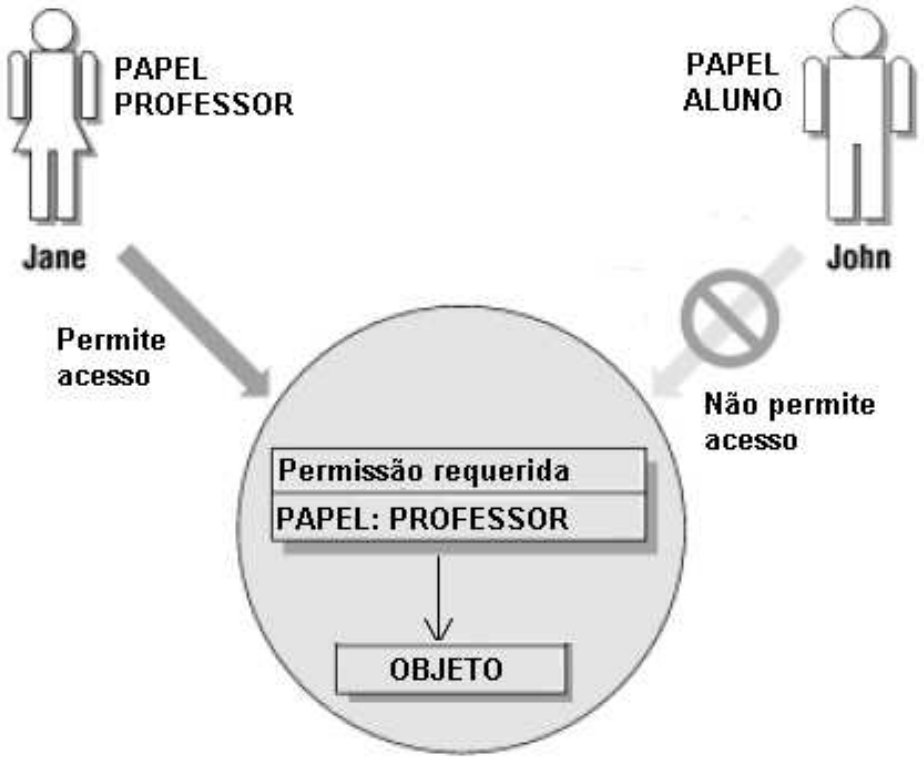

Figura 2.7: Controle de Acesso RBAC, adaptado de Russell e Gangemi (1991)

2.10 Conclusão

Este capítulo apresentou os principais fundamentos tecnológicos relacionados à segurança de sistemas computacionais, que também são utilizados na infra-estrutura de grades computacionais. Por exemplo, o mecanismo de autenticação baseado em certificados digitais é muito utilizado por grades computacionais. Entre os modelos de controle de acesso apresentado, o RBAC é o modelo mais interessante para controlar ação ou operação de usuários que será utilizado na proposta apresentada no capítulo 4.

Capítulo 3

Grades Computacionais

Este capítulo apresenta alguns conceitos importantes sobre o padrão OGSA que oferece diretrizes para construção de grades computacionais, uma visão geral da ferramenta Globus Toolkit versão 4 (GT4) e os principais conceitos de portais de grade. Os componentes de segurança da ferramenta GT4 e dos portais de grade são temas importantes no entendimento da arquitetura proposta no próximo capítulo.

Apesar dos primeiros sistemas de computação em grade (*Grid Systems*) terem sido desenvolvidos em meados dos anos 90, existem estudos que datam dos anos 60 que já antecipavam o que seria um sistema de computação em grade, como o artigo de Licklider e Taylor (1968).

Dois projetos representativos da vanguarda desse tipo de tecnologia foram o FAFNER (NPAC, 1995) e o I-WAY (DeFanti et al., 1996). Esses projetos diferem em vários aspectos, mas ambos venceram um número semelhante de obstáculos, incluindo comunicação, gerenciamento de recurso e manipulação remota de dados para conseguir trabalhar de forma eficiente e efetiva (Roure et al., 2003).

Em 1995, o FAFNER foi utilizado para fatoração de grandes números através de uma interface *web*. A fatoração de grandes números ainda é um desafio para a segurança digital e exige grande poder computacional. Por essa razão, algoritmos que permitem a fatoração em paralelo foram desenvolvidos de forma que o problema de fatoração pudesse ser distribuído aos vários contribuintes do projeto FAFNER. Assim, foi possível prover uma pequena parte do esforço de computacional para a fatoração completa do número

proposto (Roure et al., 2003). I-WAY foi outro projeto concebido no começo de 1995 com o objetivo de interligar supercomputadores (recursos) distribuídos no território dos EUA e aplicar o poder computacional criado para executar aplicações distribuídas de realidade virtual. Segundo a classificação de Roure et al. (2003), esses projetos fazem parte da primeira geração de *grid* que envolvia soluções proprietárias para compartilhar recursos de alto desempenho com a finalidade de atender a aplicações específicas.

A partir de então, surgiram outros projetos que se preocuparam com adoção de padrões e criação de um *middleware* (uma infra-estrutura entre as aplicações e o sistema operacional) para atender às necessidades da computação em grade. Por exemplo, entre projetos da segunda geração destacam-se: UNICORE (1997), Condor (1988), Legion (1997) e o Globus (1996).

O Globus Toolkit (GT) é desenvolvido e mantido pelo consórcio Globus Alliance, o qual oferece uma infra-estrutura de software que capacita a execução de aplicações que lidam com recursos computacionais heterogêneos e distribuídos. O GT é um conjunto básico de componentes que implementam os serviços básicos como segurança, localização de serviços, gerenciamento de recursos e comunicação para construção de uma grade computacional (Roure et al., 2003). Da versão 1 do GT, de 1998 para a versão 2 de 2002, os protocolos e serviços envolvidos foram alterados, mas ainda faltava um padrão para construção de grades computacionais. A ferramenta Globus Toolkit versão 3 (GT3) foi a primeira versão a adotar o padrão *Open Grid Services Architecture* (OGSA) coordenado pelo *Open Grid Forum* (OGF) [1], com o objetivo de fornecer um padrão aberto que seja uma referência para o desenvolvimento de aplicações para grades computacionais.

Vários aspectos relacionados com o desenvolvimento de grades computacionais são mantidos por entidades, tais como OGF, *Organization for the Advancement of Structured Information Standards* (OASIS) entre outras. Atualmente, essas organizações concentram e coordenam seus esforços de desenvolvimentos de padrões tecnológicos com a colaboração da indústria e da área acadêmica.

[1] Recentemente, o *Global Grid Forum* (GGF) se fundiu com o *Enterprise Grid Alliance* (EGA) e passou a ser chamado de *Open Grid Forum* (OGF).

3.1 Open Grid Services Architecture

As primeiras plataformas de computação em grade foram construídas sem a adoção de padrões que não existiam em meados dos anos 90. Nos últimos anos, com a mobilização de esforços conjuntos entre a comunidade científica e a indústria, sob a coordenação do GGF, observa-se uma forte tendência de uma Arquitetura Orientada a Serviços (AOS) na construção de padrões abertos para grades computacionais que resultou na criação da arquitetura OGSA. Essa arquitetura oferece um arcabouço para implementação dos recursos necessários para uma grade computacional (Jacob et al., 2005).

A finalidade da arquitetura OGSA é fornecer um padrão aberto que sirva de referência para o desenvolvimento de aplicações para grades computacionais. A arquitetura OGSA padroniza praticamente todos os serviços e funcionalidades que podem ser encontradas em grades computacionais, como por exemplo, submissão de tarefas, serviço de informação, serviços de transferência de dados, autenticação, autorização, etc. Ela se baseia em um conjunto de protocolos e padrões de serviço *web*, desenvolvido e mantido por várias organizações internacionais como a *World Wide Web Consortium* (W3C), *Organization for Advancement of Structured Information Standards* (OASIS), *Distributed Management Task Force* (DMTF) e *Web Services Interoperability Organization* (WS-I) (Senger et al., 2006).

O desenvolvimento de aplicações em grades computacionais em conformidade com a arquitetura OGSA foi inicialmente feita através do padrão *Open Grid Services Interface* (OGSI). A ferramenta Globus Toolkit na sua versão 3 (GT3) é a implementação de referência na aplicação desse padrão (Jacob et al., 2005). Entretanto, essa implementação utilizou algumas extensões de serviços *web* denominada OGSI, que geraria dificuldades em uma futura integração dos serviços *web* com as aplicações para grades computacionais devido à contínua evolução dos serviços *web*. A principal divergência do padrão OGSI é alteração de algumas convenções populares adotadas pela comunidade de serviços *web*. O padrão *Web Services-Resource Framework* (WSRF) surgiu posteriormente, com uma proposta de solução para o problema que aproximou a comunidade de desenvolvimento de serviços de grade (*grid services*) com a comunidade de serviços *web* (Jacob et al., 2005).

Um dos pontos centrais da arquitetura OGSA é o serviço de grade, que é um serviço *web* adaptado às condições e ao ambiente de grades computacionais. Serviços de grade apresentam certas especificidades que tiveram de ser tratadas de forma diferenciada dos serviços *web* em geral. A principal diferença é que os serviços *web* não possuem um estado associado (*stateless*), ou seja, não mantém quaisquer informações entre duas ou mais invocações. Ao contrário disso, a maioria dos serviços *grid* precisa manter algum estado ao longo de uma seqüência de invocações. Para atender a essa e outras especificidades foi criado o padrão WSRF, sucessor do OGSI, e implementado na ferramenta Globus Toolkit versão 4 (GT4). O elemento básico de construção do WSRF é o chamado WS-Resource. WS-Resource é o modelo de construção usado para representar recursos com estado usando a arquitetura de um arcabouço de serviços *web*. De acordo com o padrão WSRF: i)Um recurso tem seu estado descrito como um documento XML. ii) Um recurso tem um ciclo de vida definido. iii) Um recurso é acessível (e conhecido) por um ou mais serviços *web*. Em outras palavras, WS-Resource é a combinação de um serviço *web* e um recurso com estado (**WS-Resource = Serviço *Web* + Recurso com estado**). Ele pode ser manipulado por meio de operações de requisição e alteração de suas propriedades. O WSRF é uma série de especificações que define o padrão ou a maneira de requisitar o valor da(s) propriedade(s) ou especificar a alteração de um WS-Resource.

O WSRF define convenções para a troca de mensagens usada para interagir com o estado por meio de um arquivo descrito com a linguagem chamada de *Web Services Description Language* (WSDL). A linguagem WSDL contém definições sobre troca de mensagens entre dois lados da conversação de um serviço *web*. O padrão WSRF possui quatro especificações que define como representar, acessar, gerenciar e agrupar WS-Resources (Jacob et al., 2005):

- WS-ResourceProperties: Define como os WS-Resources são descritos pelos documentos XML e como os recursos podem ser consultados ou modificados.
- WS-ResourceLifetime: Define os mecanismos de controle do ciclo de vida dos WS-Resources.
- WS-ServiceGroup: Descreve como a coleção de serviços *web* ou WS-Resources podem ser representados e gerenciados.

- WS-Faults: Define um padrão para registro de uma falha na troca de mensagens.

A vantagem dessa convenção é construir serviços mais fáceis de usar e gerenciar. WSRF está em conformidade com o padrão *WS-Interoperability* (WS-I) *Basic Profile*, que permite interagir com qualquer serviço que suporte o padrão WSRF (Akram, 2005). Implementações de WSRF são disponibilizadas por ferramentas como GT4 (Alliance, 2007b) e WSRF.NET (Humphrey e Wasson, 2005). A Figura 3.1 ilustra o diagrama em camadas para desenvolvimento de uma solução aberta de grades computacionais.

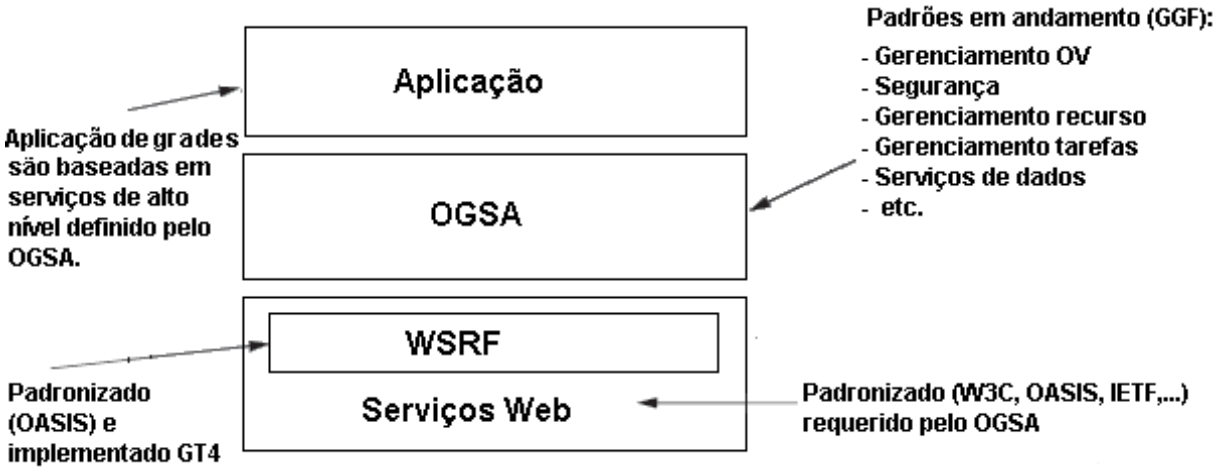

Figura 3.1: Diagrama em camadas GT4, OGSA, WSRF e serviços *web*, adaptado de Sotomayor e Childers (2006)

3.2 Globus Toolkit 4

O Globus Toolkit 4 (GT4) é uma ferramenta composta por um conjunto de serviços, bibliotecas de programação e ferramentas de desenvolvimento que provê suporte para a implementação e execução de aplicações para grades computacionais (Foster, 2006). O GT4 é composto por um conjunto de ferramentas que permite a construção de sistemas de grade de forma modular. O termo *toolkit* é devido ao fato de ele permitir a configuração e personalização de grades especializadas e aplicações. Além disso, é um "pacote" de ferramentas que se complementam, em vez de um *middleware* monolítico. É comum dizer que Globus é a implementação de referência da arquitetura OGSA conforme ilustra a Figura 3.2. GT4 é atual versão do Globus.

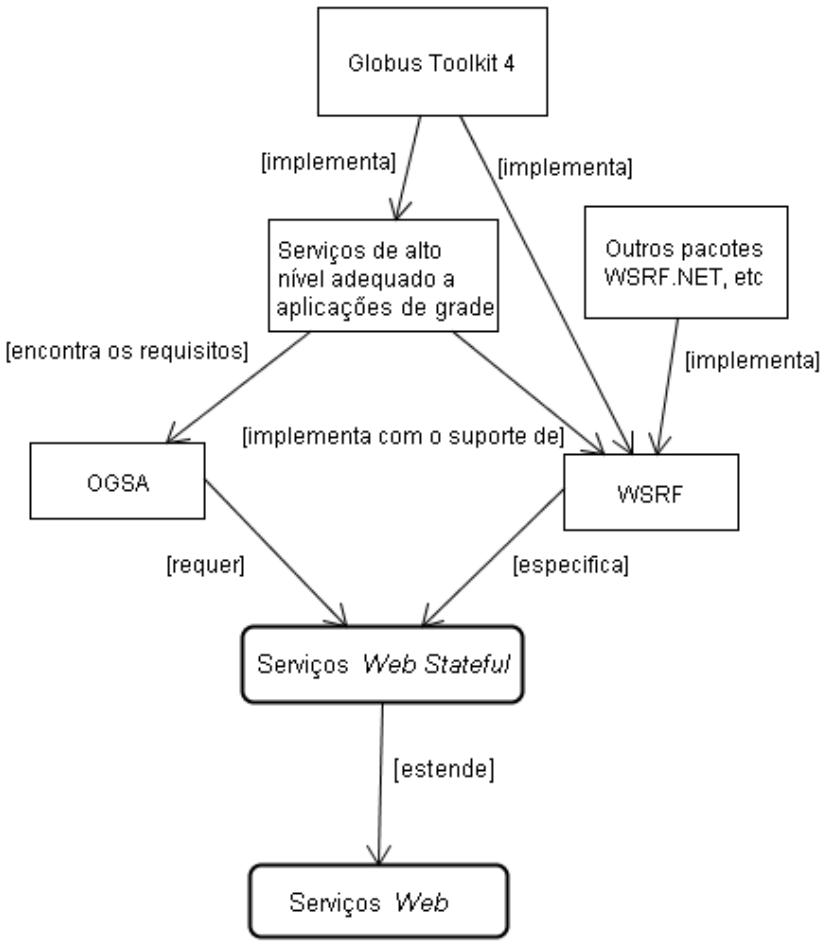

Figura 3.2: Relacionamento entre GT4, OGSA, WSRF e serviços *web*, adaptado de Alliance (2007b)

O GT4 engloba um conjunto de componentes de software aberto que podem ser classificados em cinco grupos, conforme ilustra a Figura 3.3.

O primeiro grupo provê suporte a um conjunto de requisitos relacionados a segurança em grades computacionais, incluindo a verificação das identidades de usuários e de recursos, a proteção a troca de mensagens, controle de permissões de acesso a recursos e serviços e a implementação de políticas de acesso. O grupo da infra-estrutura de segurança do GT4 é conhecido como *Grid Security Infrastructure* (GSI), é composto pelos seguintes componentes:

- *Credential Management*: É o componente que disponibiliza a ferramenta SimpleCA e MyProxy. SimpleCA é uma versão simplificada de uma autoridade certificadora baseada na funcionalidade da ferramenta de código aberto OpenSSL CA (Cox et al., 2007). MyProxy é um repositório *on-line* de credenciais usado pela comunidade

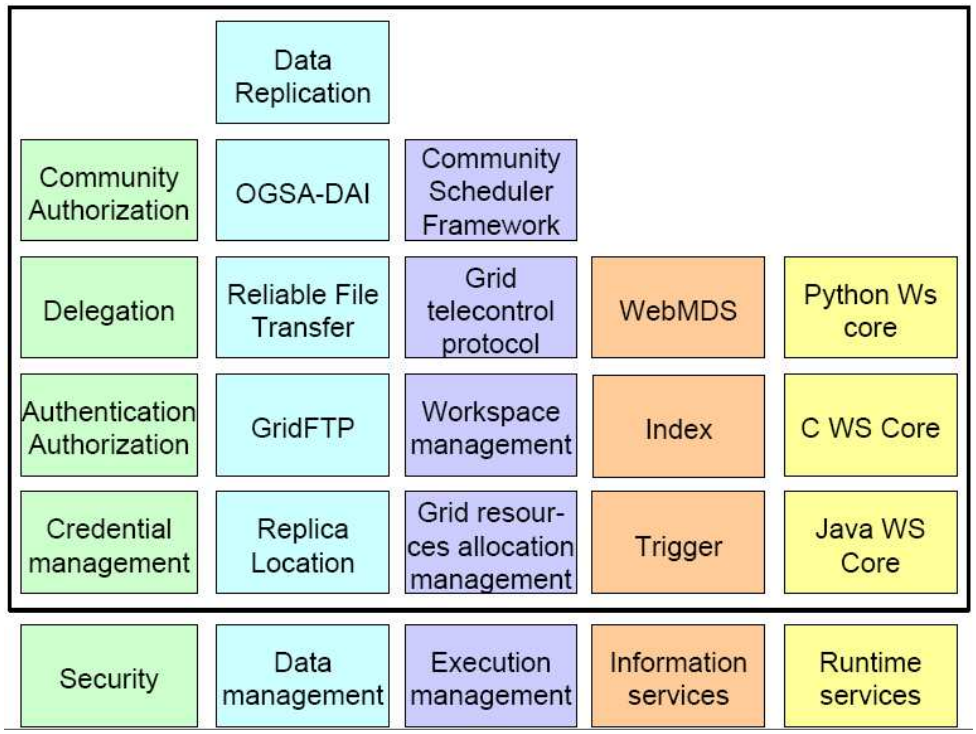

Figura 3.3: Arquitetura do GT4 (Jacob et al., 2005)

de usuários da grade para gerenciamento de suas credenciais baseada em uma infra-estrutura de chaves públicas.

- *Authentication and Authorization*: Esse componente fornece diversos mecanismos de autenticação de identidades e controle de acesso aos recursos e aos serviços disponibilizados na grade.

- *Delegation*: Permite delegar as credenciais para outras entidades (contêiner ou serviço), para agir em seu nome com as mesmas permissões de acesso. O componente também fornece suporte à renovação de credenciais.

- *Community Authorization Service* (CAS): Permite que uma organização virtual (OV) expresse suas políticas de controle de acesso aos recursos, distribuídos ao longo de vários locais (sítios, domínios).

O segundo grupo trata do acesso e do gerenciamento de dados na grade. Esse grupo disponibiliza uma série de mecanismos de acesso a dados para atender a essas funcionalidades, tais como:

- *Replica Location Service* (RLS): Esse componente provê um serviço de registro e acesso à informação sobre a localização de réplicas de arquivos e bases de dados

na grade. Sua implementação possibilita um controle escalável e eficiente sobre um conjunto muito grande de réplicas, como ficou comprovado por exemplo, através do experimento científico, *Laser Interferometer Gravitational Wave Observatory* (LIGO) que extensivamente replica dados e armazena mais de 40 milhões de arquivos distribuídos em 10 localidades diferentes (Chervenak et al., 2005).

- *GridFTP*: Similar a qualquer serviço *File Transfer Protocol* (FTP). Esse serviço disponibiliza mecanismos de transferência confiável, segura e de alto desempenho (tanto transferências entre memória para memória quanto para movimentações de dados entre disco para disco) (Foster, 2006). GridFTP utiliza o mecanismo básico de segurança do Globus para autenticar os clientes. O serviço GridFTP é utilizado por outros serviços do Globus como RFT e DRS comentados a seguir. O protocolo GridFTP inclue outras características como: transferências paralelas, transmissão parcial de dados ou mediar a transferência entre dois servidores de dados. Ele também, pode intermediar uma operação entre um cliente e um servidor convencional de FTP. Durante experimentos científicos chegou a alcançar 27 Gbit/s sobre redes de longa distância (Foster, 2006).

- *Reliable File Transfer* (RFT): Esse serviço implementa a gerência confiável de múltiplas transferências de arquivos que utilizam o GridFTP. Por exemplo, existe experimento para orquestração de transferência de um milhão de arquivos entre dois repositórios da área de astronomia (Senger et al., 2006).

- *Data Replication Service* (DRS): Esse componente implementa os serviços de RLS e GridFTP para prover o gerenciamento de replica arquivos registrados no RLS, utilizando o serviço RFT (Senger et al., 2006).

- *OGSA-Data Access and Integration* (OGSA-DAI): É um conjunto de ferramentas que fornece suporte ao acesso a bases de dados relacionais e XML (Foster, 2006).

O terceiro grupo disponibiliza os seguintes componentes para gerenciar alocação de recursos e da execução de tarefas na grade:

- *Grid Resources Allocation Manager* (GRAM): É o serviço que provê uma interface *web* que permite iniciar, monitorar e gerenciar a execução arbitrária em computado-

res remotos. Essa interface permite que os clientes especifiquem o tipo e quantidade de recursos necessários para execução de uma tarefa, a transferência de arquivos de entrada e arquivos executáveis aos locais de execução, parâmetros, credenciais necessárias e as necessidades de persistência de dados das tarefas (Foster, 2006).

- *Community Scheduler Framework* (CSF): Esse componente provê uma simples interface para diferentes escalonadores de tarefas (*resource schedulers*) como *Portable Batch System* (PBS), Condor, *Load Sharing Facility* (LSF) e *Sun Grid Engine* (SGE) (Sotomayor e Childers, 2006).

- *Workspace Management Service* (WMS): Provê a alocação dinâmica de espaços de trabalho em máquinas remotas para a execução de tarefas (Senger et al., 2006).

- *Grid TeleControlProtocol* (GTCP): É o serviço utilizado para gerenciamento de instrumentação remota de dispositivos especiais, como microscópios e dispositivos de experimentação sobre terremotos utilizados pelo projeto NEES (Foster, 2006).

O quarto grupo é o serviço de informação, comumente chamado de *Monitoring and Discovery System* (MDS), inclui um conjunto de componentes para monitoramento e descoberta de recursos na grade. Os serviços de descoberta permitem identificar recursos e serviços em um ambiente com alto grau de heterogeneidade que é o ambiente presente em grades computacionais. Os serviços de monitoramento simplifica a tarefa de acompanhar o estado atual ou diagnosticar problemas ocorridos com seus serviços ou recursos (Senger et al., 2006). Os seguintes componentes fazem parte desse grupo:

- *Index*: Esse serviço permite coletar informação sobre outros recursos ou serviços de grade que são disponibilizadas como propriedades, e então são publicadas para que sejam consultadas pelos clientes. Ele coleta informações de diferentes fontes de dados através do padrão WS-ResourceProperties e do serviço WS-BaseNotification. Qualquer serviço pode publicar a informação de acordo com a especificação WSRF que pode ser indexada através desse serviço (Chakrabarti, 2007).

- *Trigger*: Esse é o serviço que permite comparar as informações coletadas com um conjunto de condições, que uma vez alcançadas, permite disparar ações pré-

estabelecidas. Por exemplo, um administrador pode necessitar receber um *e-mail*, caso um nó falhe ou quando o servidor alcance seu limite.

- *WebMDS*: Oferece uma interface *web* capaz de exibir informações coletadas pelos outros serviços. Ela pode ser usada como uma interface mais amigável para o serviço Index (Chakrabarti, 2007).

O quinto e último grupo, inclui uma série de componentes comuns para execução (contêineres), desenvolvimento e implantação de novos serviços em linguagem Java, C ou Python. Para cada uma dessas linguagens existe um contêiner (repositório onde os serviços podem ser acessados) que provê todo o ambiente de execução para os serviços *web*. Muitos dos componentes do GT4 são baseados em serviços *web*. Para que esses componentes sejam executados precisam estar dentro de um contêiner de serviços *web* e, desta forma, vão poder atender às requisições de diferentes clientes. O GT4 se baseia fortemente na linguagem Java, sendo atualmente poucos recursos que ainda se mantém na linguagem C.

3.2.1 Certificados Proxy

A infra-estrutura de segurança do GT4 é baseada em criptografia de chave pública, certificados X.509, e o estabelecimento de um canal seguro através do protocolo de comunicação *Secure Socket Layer* (SSL). Essas tecnologias são utilizadas na segurança da grade, não somente por serem tecnologias bem conhecidas e já padronizadas pela comunidade, mas também por permitirem o estabelecimento de uma relação de confiança baseada em certificados X.509. Isso permite que uma entidade possa confiar em outra entidade de diferente domínio administrativo (Autoridade Certificadora de outra organização) sem precisar obrigar o resto da sua organização a confiar nela também. Em outras palavras, não requer a reciprocidade pela Autoridade Certificadora confiada. A flexibilidade foi um fator decisivo para escolha entre esse modelo de confiança e outros mecanismos de autenticação para uso em grades (Welch et al., 2004). Por exemplo, um mecanismo de autenticação baseado no protocolo Kerberos exige que seja estabelecida uma relação de confiança com todos os diferentes domínios, de forma que as organizações possam permitir

a autenticação entre domínios, o que pode tornar o processo de gerenciamento bastante pesado.

Entretanto, existem casos não atendidos por esse modelo de confiança quando utilizado somente certificados X.509 padrão. Por exemplo, o modelo baseado em *proxies* permite que uma entidade (serviço, recurso ou usuário) passe seu privilégio para outra, por um breve período de tempo e sem que a senha tenha que ser digitada novamente, a cada vez que um recurso ou serviço precise ser acessado (Welch et al., 2004). A solução baseada no uso de certificados *proxy* acrescenta duas características adicionais ao certificado X.509 padrão: autenticação única e delegação.

Certificados *proxy* usam o mesmo formato dos certificados digitais descritos na seção 2.3, o que permite associar uma chave pública para um sujeito da mesma forma usada por certificados X.509 de chave pública. Em razão de utilizar o mesmo formato que os certificados X.509 de chave pública, permite ser utilizada por protocolos e bibliotecas como se fosse um certificado X.509 padrão, reduzindo o custo de desenvolvimento de aplicações. Essa técnica permite que uma entidade possuidora de um certificado padrão X.509 delegue privilégios para outra entidade a qual pode não possuir nenhuma credencial X.509 no momento da delegação. A delegação pode ser executada dinamicamente, sem a assistência de terceiros, e pode ser limitada por um conjunto arbitrário de privilégios da entidade delegada. Uma vez obtido o certificado *proxy*, ele é usado pelo seu proprietário para autenticar e estabelecer conexão segura com outros domínios da mesma forma que os certificados X.509 normais. Certificados *proxy* foram utilizados desde as primeiras versões do GSI, e conseqüentemente, passaram por um processo de refinamento até a padronização feita pelo grupo PKIX do IETF. A Figura 3.4 resume as diferenças entre o certificado X.509 padrão e o *proxy*.

O campo *Emissor* do certificado X.509 *proxy* é identificado pela chave pública do certificado ou por outro certificado *proxy*. Essa opção permite que o certificado seja criado dinamicamente pelo proprietário do certificado, sem a intervenção da AC (Autoridade Certificadora), ou seja, sem passar pelo custoso processo de geração de certificado X.509 feito pela mesma. O campo *Sujeito* do certificado X.509 *proxy* permite a definição do proprietário dentro do escopo de nomes do emissor do certificado. A restrição de escopo

Atributo do Certificado	X.509	X.509 Proxy
Emissor/Assinante	Autoridade Certificadora	Certificado de chave pública ou Certificado de outro *proxy*
Sujeito	Definido pela Autoridade Certificadora	Definido no espaço de nomes do proprietário do certificado
Delegação do Emissor	Nenhuma	Define as condições de direitos delegados
Par de Chaves	Par de chaves única	Par de chaves única

Figura 3.4: Comparação entre certificados X.509 e *proxy*

deve ser feita para que os certificados sejam únicos. A chave pública do certificado *proxy* é distinta da chave pública do emissor e pode ter diferentes propriedades, como por exemplo, as chaves podem ter tamanhos diferentes.

3.2.2 Autenticação Única

O certificado *proxy* permite a autenticação única (conhecida como *Single Sign On* - SSO) pois dispensa a digitação freqüente de senha para provar a identidade de uma entidade que pode ser um serviço, recurso ou usuário. Em outras palavras, permite reduzir o número de vezes em que o usuário precisa acessar sua chave privada.

O processo de autenticação do Globus Toolkit é ilustrado na Figura 3.5. Inicialmente, um novo par de chave (composto de uma chave pública e uma privada) é criado e usado para gerar uma requisição padrão para obter um certificado X.509. No próximo passo, o usuário utiliza sua chave privada para criar e assinar o certificado *proxy*. O certificado *proxy* associa uma nova chave pública com um novo nome e delega alguns ou todos os privilégios do usuário para esse novo nome. Os certificados *proxy* têm um tempo de vida ou exposição muito menor quando comparado com certificados X.509 padrão. Ao contrário dos certificados X.509 padrão que são protegidos com senha, os certificados *proxy* são protegidos somente pelas permissões de arquivos do sistema operacional.

3.2.3 Delegação

A delegação permite que uma entidade se conecte a qualquer recurso em nome de outra entidade. A delegação no GT4 pode ser feita sobre um canal de comunicação seguro, conforme ilustra a Figura 3.6. Inicialmente, os dois elementos envolvidos na

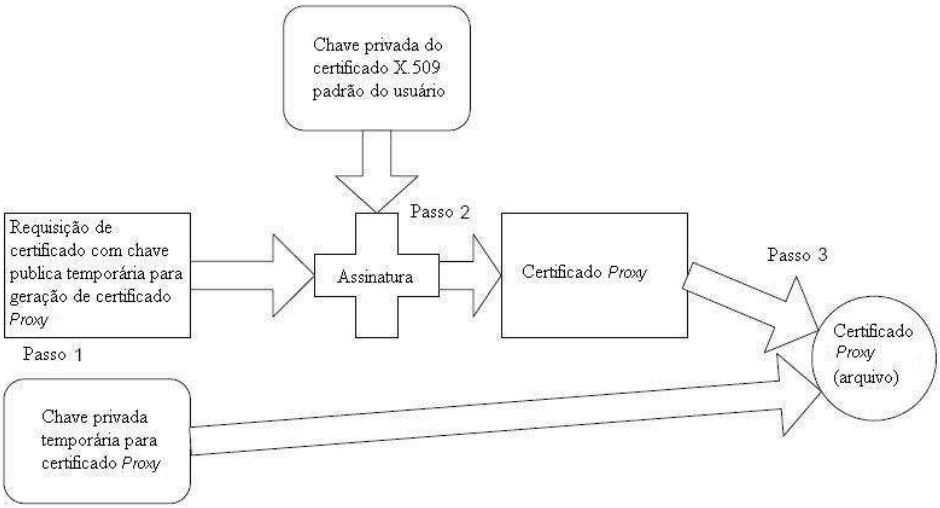

Figura 3.5: Processo de autenticação única, adaptado de Welch et al. (2004)

delegação negociam um canal de comunicação seguro através de um processo chamado de autenticação mútua onde ambos verificam a identidade um do outro. O delegador (servidor A) faz uso do seu certificado *proxy* existente e o delegado (servidor B) utiliza seu certificado de chave pública não ilustrado na Figura 3.6. Então, no passo seguinte, o delegado (aquele que recebe a delegação) cria um par de chaves e uma requisição de certificado. O delegado, então, envia a requisição do certificado ao delegador através de um canal seguro semelhante ao processo usado para requisitar certificados de uma Autoridade Certificadora. Ao receber a requisição do certificado, o delegador assina com sua chave privada e cria um novo certificado *proxy*. O delegador envia o certificado sobre o canal seguro de rede. Ao receber o certificado, o delegado armazena em um local seguro, junto com a chave privada criada no início do processo.

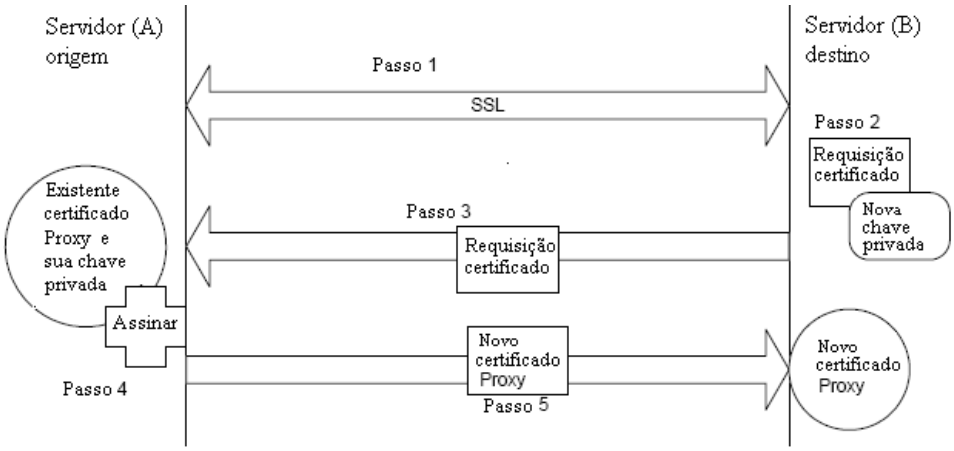

Figura 3.6: Delegação de um certificado *proxy* sobre um canal seguro, adaptado de Welch et al. (2004)

Certificados *proxy* podem também ser criados com a finalidade de delegar privilégios para uma outra entidade com a qual foi estabelecido um canal de comunicação, sem a necessidade de troca de nenhuma chave privada entre eles. Esse processo de delegação exige o transporte com proteção de integridade das mensagens para impedir adulterações no conteúdo das mensagens. Entretanto, não exige que as mensagens sejam criptografadas, pois nenhuma informação confidencial é trocada (Welch et al., 2004).

3.2.4 MyProxy

Myproxy é um sistema cliente-servidor no qual os clientes podem armazenar credenciais sob o controle de políticas de acesso em um repositório *on-line* para posteriormente poder recuperá-las, conforme mostrado na Figura 3.7. MyProxy utiliza certificados *proxy* (Tuecke et al., 2004) para delegar e recuperar credenciais de um repositório sem exportar a chave privada. Normalmente, o usuário armazena uma credencial chamada de longa duração (certificado X.509 padrão) com validade de semanas ou anos dentro do repositório e recupera uma credencial de curta duração (certificado *proxy*) com validade de um dia ou menos para acesso a grade. Isso permite que a credencial de longa duração permaneça protegida pelo servidor MyProxy. Ele é um produto de software *open-source* que foi desenvolvido para atender às necessidades de gerenciamento de credenciais para computação em grade utilizando o *middleware* Globus Toolkit. Ele é utilizado pela comunidade de grades computacionais desde a sua primeira versão lançada em 2000 (Novotny et al., 2001), tornando-se um importante componente de grandes projetos *grid*, tais como: NEESgrid, EU DataGrid e NASA Information Power Grid (IPG). Tanto o componente cliente quanto o servidor foram desenvolvidos em linguagem C usando a biblioteca GSS-API (Linn, 1997) fornecida pelo Globus Toolkit, a qual utiliza a biblioteca OpenSSL para implementar o protocolo TLS e tratar os certificados X.509. O componente cliente pode ser também desenvolvido em Java utilizando a ferramenta CoG Kit (Laszewski et al., 2001).

O Myproxy permite recuperar, armazenar, obter e remover uma credencial *proxy* através do protocolo MyProxy que pode ser descrito como segue: (i) o cliente estabelece uma conexão TCP para o servidor MyProxy e inicializa os requisitos necessários do protocolo

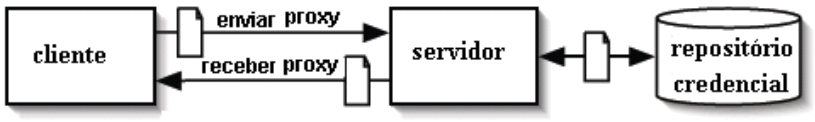

Figura 3.7: Visão geral do MyProxy, adaptado de Basney et al. (2005)

TLS. O servidor deve ser autenticado (reconhecido) do ponto de vista do cliente que utiliza um nome qualificado, *Fully-Qualified Host Name* (FQHN). O cliente também poderia ser autenticado em relação ao servidor através do protocolo TLS, mas para permitir que clientes sem credenciais X.509, obtenham credenciais do servidor MyProxy, não é exigida autenticação do lado cliente. (ii) O cliente e o servidor criam um canal de comunicação seguro entre eles. Uma vez estabelecido um canal seguro, o cliente envia um indicador que não está delegando a credencial nesse momento, como parte do protocolo GSS-API (Linn, 1997) do Globus Toolkit. (iii) Então, a comunicação prossegue com as trocas de mensagens que fazem parte do protocolo MyProxy.

Segundo Basney, Humphrey e Welch (Basney et al., 2005) destacam quatro aplicações do MyProxy envolvendo computação em grade, a saber:

- Mobilidade: É a facilidade dos usuários obterem suas credenciais quando necessárias independente da sua localização dentro da rede. Ou seja, o usuário não precisa iniciar sua sessão na grade sempre do mesmo computador, nem ser obrigado a copiar sua credencial X.509 para cada uma das diferentes localidades onde ele precise de acesso. Os usuários do projeto TeraGrid são um exemplo de mobilidade, armazenam suas credenciais em um servidor MyProxy, protegidos por uma senha e utilizam clientes MyProxy para obter as credenciais nas diferentes localidades nas quais estão distribuídos os recursos dentro dos Estados Unidos.

- Renovação: É o mecanismo de renovação da validade das credenciais de curta duração para atender à execução de tarefas de longa duração. Normalmente, é difícil prever quanto tempo uma tarefa levará para ser concluída em uma grade e delegar credenciais de longa duração para tarefas de longa duração é um risco para segurança. O EU DataGrid desenvolveu uma solução para esse problema utilizando MyProxy. Os usuários obtém delegação de credenciais de curta duração para suas

tarefas através do servidor MyProxy. Um serviço chamado de WMS (Workload Management System) monitora as tarefas e renova as credenciais dos usuários quando necessário junto ao MyProxy.

- Delegação: É o mecanismo de adicionar delegação para protocolos existentes que não suportam esse recurso. Um dos principais exemplos é o portal para grade que provê uma interface *web* compatível com o padrão dos navegadores *web*. Mas, para interagir com os recursos da grade em nome de um usuário, o portal para grade precisa acessar a credencial do usuário. Entretanto, o padrão dos navegadores *web* não suporta delegação. O problema é resolvido armazenando as credenciais dos usuários em um servidor MyProxy e permitindo que a aplicação do portal interaja com o mesmo. Vale destacar que as mais importantes soluções para integrar o portal e a grade computacional tem alguma interface com o MyProxy, isso inclui projetos como GridSphere (Novotny et al., 2006) e OGCE (Pierce et al., 2005).

- Registro: É o mecanismo de integrar MyProxy e uma AC de forma a diminuir o gerenciamento dos usuários. Quando um novo usuário é criado em um domínio administrativo, um serviço de contas pode requisitar um certificado em nome do usuário para uma Autoridade Certificadora e armazenar as credenciais diretamente no repositório configurando uma senha inicial para ele. O usuário ao ser notificado pode utilizá-la para ter acesso a sua nova conta na grade, recuperando-a de um servidor MyProxy. A versão mais recente do MyProxy inclui scripts que facilitam a integração com a ferramenta SimpleCA, incluída no pacote do Globus Toolkit.

3.2.5 Soluções para a Autorização

Um dos princípios mais importantes do modelo de segurança do Globus é que somente os usuários autorizados estejam aptos a utilizar os recursos. O mecanismo mais primitivo de autorização oferecido pelo GT4 é feito através do arquivo *grid-mapfile*. Esse arquivo é criado localmente pelo administrador para cada computador do *grid* e especifica quais computadores ou usuários estão autorizados a ter acesso aos recursos desse computador. O uso do arquivo *grid-mapfile* é mantido por questão de compatibilidade com a versão 2

do GT, e associa um determinado usuário do sistema operacional a um identificador X.509 que aparece no seu certificado. O mecanismo de autorização por meio de *grid-mapfile* não tem a escalabilidade e flexibilidade necessárias para atender grandes sistemas distribuídos. Um exemplo de uma entrada no arquivo *grid-mapfile* é listado abaixo:

```
"/O=Grid/OU=BR/OU=Unisantos/OU=br.biz/CN=Vanderlei Costa" vfcosta
```

Cada linha do grid-mapfile é a representação do mapeamento da informação contida no certificado X.509 e para um usuário local do sistema operacional.

Outro mecanismo de autorização desenvolvido como parte do GT é o *Community Authorization Services* (CAS) (Pearlman et al., 2002) que é responsável pelo gerenciamento de políticas de acesso distribuídas dentro de comunidades virtuais. Uma comunidade pode conter várias outras comunidades onde cada um dos participantes pode consumir ou oferecer recursos. Expressar políticas em termos de relações de confiança entre produtores e consumidores de recursos gera problemas de escalabilidade, flexibilidade, expressividade e falta de uma hierarquia de políticas (Pearlman et al., 2002). Para resolver esses problema, o GT introduz o servidor CAS que é responsável pelo gerenciamento das políticas que governam os acessos aos recursos de comunidades. O serviço CAS pode ser visto como um serviço autorizado pela comunidade para permitir acesso dos usuários aos recursos. O CAS implicitamente assume que usuários e recursos da comunidade concordam em ter o CAS como serviço de autorização.

A Figura 3.8 representa um usuário, membro de uma comunidade, solicitando ao servidor CAS uma requisição de uma lista de competências (*capability*). Essa lista de competências permite que ele execute um conjunto de ações específicas. Se a requisição está de acordo com a política definida pela comunidade, o servidor CAS irá delegar à lista de competência apropriada para o usuário. De posse dessa autorização, o usuário pode usar essa lista de acordo com os direitos permitidos, ou delegá-lo a terceiros. Para poder acessar o recurso, o usuário deve apresentar a lista de competência obtida com o servidor CAS ao provedor de recurso que verifica se a lista apresentada pelo usuário e as políticas locais definidas para o recurso, permitem o acesso ao mesmo. Caso positivo, é permitido o acesso ao recurso.

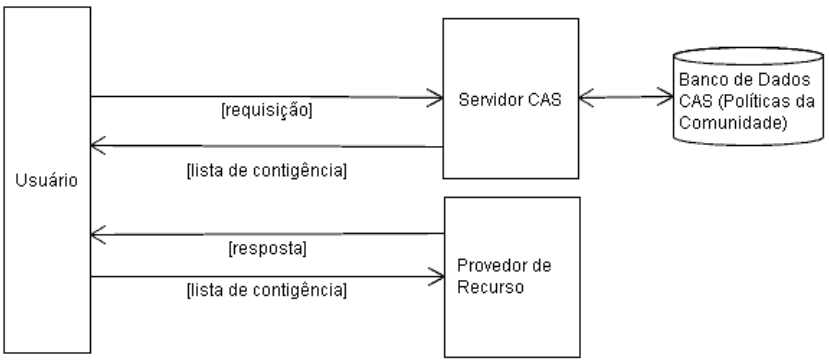

Figura 3.8: Funcionamento do CAS, adaptado de Chakrabarti (2007)

O CAS está sendo desenvolvido para atender às necessidades de segurança das comunidades virtuais e possui algumas limitações. O CAS não suporta acesso anônimo, e não utiliza certificado de atributo, mas exige um novo certificado *proxy* com extensões para conter atributos. Essa restrição limita a utilização a serviços compatíveis com o CAS. A versão corrente do CAS não oferece o recurso de autorização para serviços *web*, sendo suportado apenas pelo serviço GridFTP do GT4.

O GT4 introduziu um poderoso e flexível arcabouço de autorização que permite a implementação de um mecanismo de autorização mais sofisticado, conforme ilustra a Figura 3.9. O arcabouço de autorização possui uma série de interceptadores de mensagens de invocação de serviços para processar cada mensagem gerada antes de alcançar a aplicação. Dois tipos de interceptadores são de interesse da perspectiva de autorização: o *Policy Decision Point* (PDP), que é um componente que toma a decisão de autorização, e o *Policy Information Point* (PIP), que representa o componente que coleta a informação de atributos necessários para a tomada de decisão. Nessa arquitetura, o GT4 é representado como um *Policy Enforcement Point* (PEP), sendo o responsável pela aplicação da decisão tomada pelo PDP e pelo repasse da informação coletada pelo PIP para o PDP.

O GT4 também faz uso da linguagem *Security Assertion Markup Language* (SAML) para permitir uma chamada a um serviço externo de decisão de autorização (PIP ou PDP). Esse mecanismo permite integração com soluções de PIP ou PDP desenvolvidas por terceiros, como por exemplo o projeto GridShibPermis (D.W.Chadwick et al., 2006). O padrão SAML (OASIS, 2005) consiste de um conjunto de especificações e esquemas XML que definem uma forma padrão para criar, trocar e interpretar asserções (conjunto de afirmações) de segurança entre entidades de uma aplicação distribuída.

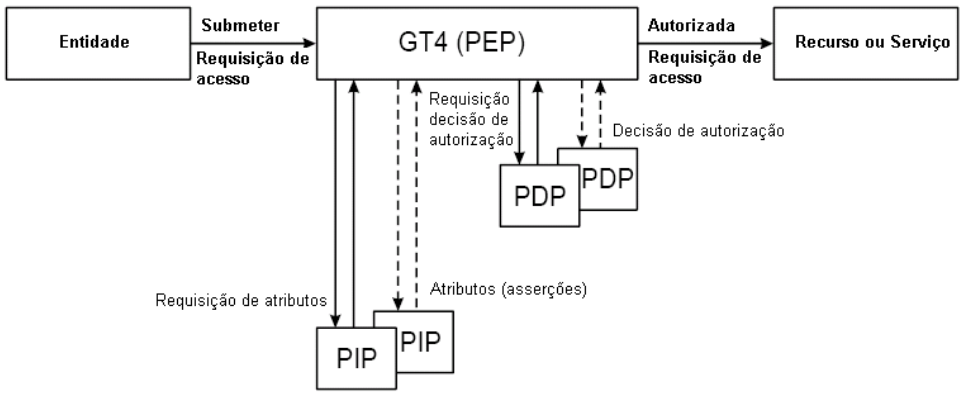

Figura 3.9: Arquitetura de autorização do GT4, adaptado de D.W.Chadwick et al. (2006)

Atualmente, existem alguns projetos representativos que utilizam atributos baseados no arcabouço de autorização do GT4, como por exemplo: o PERMIS (Chadwick e Otenko, 2003), VOMS (Alfieri et al., 2003) e AKENTI (Thompson et al., 2003).

Vale destacar que várias questões de autorização no Globus Toolkit ainda estão no campo da pesquisa. O Globus implementa um mecanismo de autenticação baseado em certificados padrão X.509, bastante consolidado, que garante a autenticidade dos usuários através da emissão de certificados digitais. Entretanto, somente a autenticidade de uma entidade não é suficiente para garantir a segurança de usuários, recursos e serviços. A elaboração de mecanismos de autorização que sejam suficientemente escaláveis e flexíveis para atender a um número flutuante de recursos, usuários e políticas que são criados ou alterados dinamicamente, ainda constitui um desafio para a pesquisa, atualmente.

Os principais projetos ligados ao desenvolvimento de soluções para autorização são descritos a seguir:

- VOMS (Alfieri et al., 2003): Foi desenvolvido pelo EDG (European Data Grid), como parte dos projetos DataGrid e DataTag para atender às necessidades de segurança de uma organização virtual. O VOMS utiliza certificado *proxy* padrão, e inclui informações de usuário, servidor e período de validade, em uma extensão (campo extra) do certificado *proxy* de forma que ainda possa ser utilizado por outros serviços de grade. Um atributo no VOMS é composto de três diferentes elementos: grupo, papel e lista de competência. VOMS permite um mecanismo baseado em papéis onde um usuário pode ter múltiplos papéis em múltiplas organizações virtuais (Chakra-

barti, 2007). Um usuário VOMS pode ser membro de qualquer grupo que faz parte de sua hierarquia. Uma lista de competências é usada para descrever características especiais do usuário (Alfieri et al., 2005). A Figura 3.10 ilustra o mecanismo de autorização VOMS. Um usuário e o servidor VOMS se autenticam mutuamente através dos seus certificados utilizando a API padrão do Globus Toolkit. O usuário faz uma requisição assinada para o servidor VOMS, que verifica a identidade do usuário e a sintaxe do pedido. O servidor VOMS retorna para o usuário a informação requerida e assinada, cuja estrutura contém uma lista de grupos, papéis e uma lista de contingências (*capabilities*) chamada de pseudo-certificado. O usuário verifica a validade da informação recebida. Opcionalmente, um usuário pode repetir este processo para outros servidores VOMS. Então, o usuário cria o certificado *proxy* que contém toda a informação recebida do servidor VOMS. A partir desse ponto, o usuário pode executar uma chamada a um provedor de recurso, que será processado pelo serviço *gatekeeper* do Globus Toolkit que verifica a validade do certificado *proxy* e utilizando o módulo *Local Credential Authorization Service* (LCAS), a informação de autorização é extraída do certificado. A informação de autorização combinada com as políticas locais produz uma decisão de autorização. LCAS é o componente responsável pela tomada decisão de autorização no VOMS (Alfieri et al., 2005).

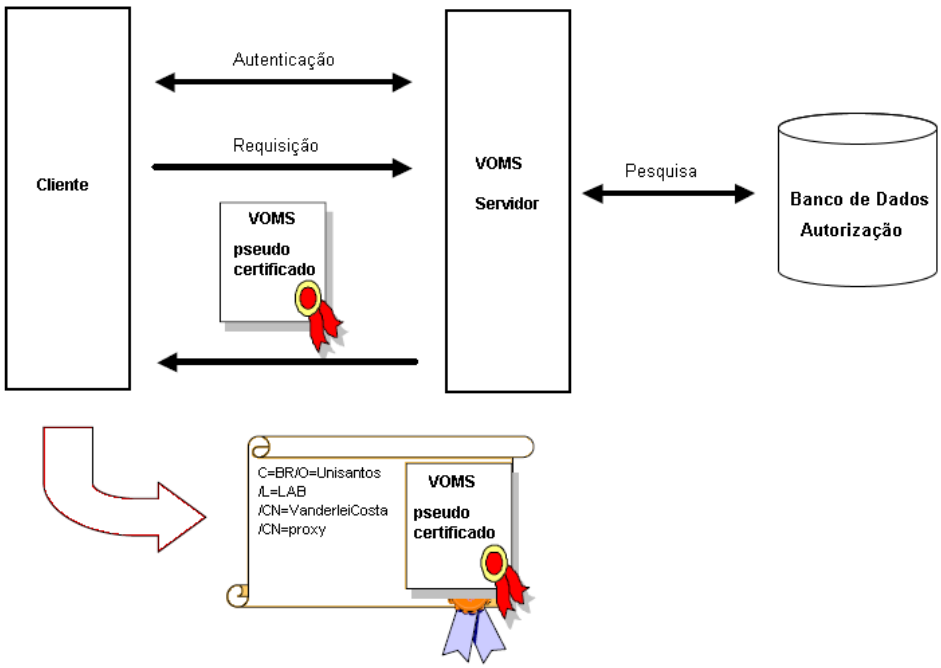

Figura 3.10: Sistema de autorização VOMS, adaptado de Alfieri et al. (2005)

- PERMIS (Chadwick e Otenko, 2003): *Privilege and Role Management Infrastructure Standards Validation* (PERMIS) é infra-estrutura de autorização financiado pela Comissão Européia, envolvendo membros de Barcelona (Espanha), Bologna (Italia) e Salford (Reino Unido). Três diferentes problemas comerciais de cada uma dessas cidades motivaram o desenvolvimento do PERMIS. Bologna queria permitir que seus arquitetos pudessem recuperar e atualizar mapas locais com a intenção de planejar a ocupação urbana da cidade, futuramente. Em Barcelona, o problema era permitir que as locadoras de carros pudessem ter acesso *on-line* ao banco de dados de controle de estacionamento da cidade. Dessa forma, as locadoras poderiam verificar os carros com cartões vencidos, identificar o motorista, normalmente estrangeiro, e transferir a multa em caso de violação. Em Salford, o problema era manter uma aplicação de licitação eletrônica, onde somente usuários autorizados poderiam submeter propostas. O ponto comum nesses três problemas é a necessidade de um mecanismo de autorização que é proposta do PERMIS (Chadwick e Otenko, 2003). Para atender às necessidades dessas diferentes aplicações a infra-estrutura de gerenciamento do PERMIS está baseada no gerenciamento de privilégios (IGP). Dentro do IGP, o direito de acesso é baseado em atributos de privilégios definidos em extensões do certificado X.509 padrão. Cada atributo dentro de um certificado de atributos descreve um ou mais permissões do usuário. Quando um usuário solicita acesso a um recurso, os atributos contidos no seu certificado serão lidos, e será verificado se a ação do usuário é permitida ou não.

  A Figura 3.11 ilustra o cenário de funcionamento do PERMIS. Um usuário solicita acesso a certo serviço ou recurso para um gateway que o direciona ao PERMIS. O PERMIS não faz nenhuma consideração sobre a autenticação empregada, deixando a definição para o domínio responsável pelo recurso.

  PERMIS é um mecanismo de autorização baseado em certificados de atributos e controle de acesso RBAC. O PERMIS permite acesso aos recursos baseado em uma política de controle de acesso ao recurso. Seus dois principais componentes são: *Access Control Enforcement Function* (AEF) e *Access Control Decision Function* (ADF). A comunicação entre a AEF e a ADF é feita através da PERMIS IGP API.

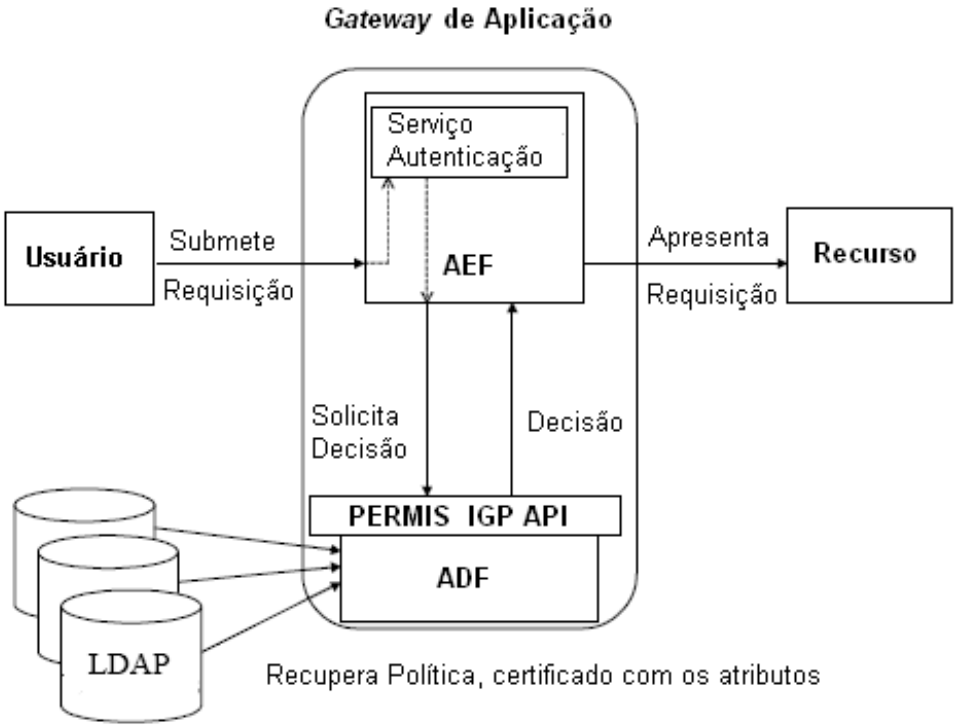

Figura 3.11: Infra-estrutura de autorização PERMIS, adaptado de Chadwick e Otenko (2003)

ADF acessa um servidor LDAP para obter as políticas que devem ser fornecidas ao AEF para tomar a decisão sobre o acesso. Os certificados são armazenados em um servidor LDAP que os disponibiliza publicamente (Chakrabarti, 2007).

- AKENTI (Thompson et al., 2003): É um exemplo de uma infra-estrutura de autorização no nível de recurso, desenvolvida no Lawrence Berkeley National Laboratory (LBNL). O AKENTI consiste de recursos distribuídos acessados por usuários que podem estar geograficamente distribuídos e em diferentes domínios administrativos. O modelo de autorização do AKENTI consiste de recursos que são acessados por usuários através de um mediador (*Policy Enforcement Point* - PEP). Esses usuários conectam ao mediador de recursos apresentando seus certificados X.509 como mecanismo de autenticação. A comunicação segura dos clientes com o PEP utiliza o protocolo SSL. As restrições de acesso aos recursos são expressas como um conjunto de certificados assinados, geralmente armazenados na mesma máquina do PEP (Thompson et al., 2003). Esses certificados expressam os atributos que um usuário deve possuir para obter os direitos de acesso aos recursos. No momento de acesso a um recurso, o usuário faz uma solicitação ao mediador (PEP) apresen-

tando suas credenciais. O mediador solicita ao servidor AKENTI quais os acessos permitidos para o usuário. Então, o servidor AKENTI pesquisa todos os certificados relevantes ao usuário, verifica se cada um dos certificados é assinado por uma entidade confiável, avalia os certificados e retorna a decisão do acesso ao PEP. Então, o PEP aplica a decisão tomada pelo servidor AKENTI, conforme ilustra a Figura 3.12. As políticas são expressas em XML e armazenadas em três tipos de certificados assinados (Chakrabarti, 2007). i) Certificados de políticas (*Policy Certificates*) que específica a localização da autoridade do recurso que geralmente contém a URL para a localização do certificado de atributos ii) Certificados de condição de uso (*Use Condition Certificates*) que contém as restrições que controlam o acesso ao recurso. iii) Certificado de atributos (*Attribute Certificates*) que especifica os atributos utilizados nos certificados de condição de uso.

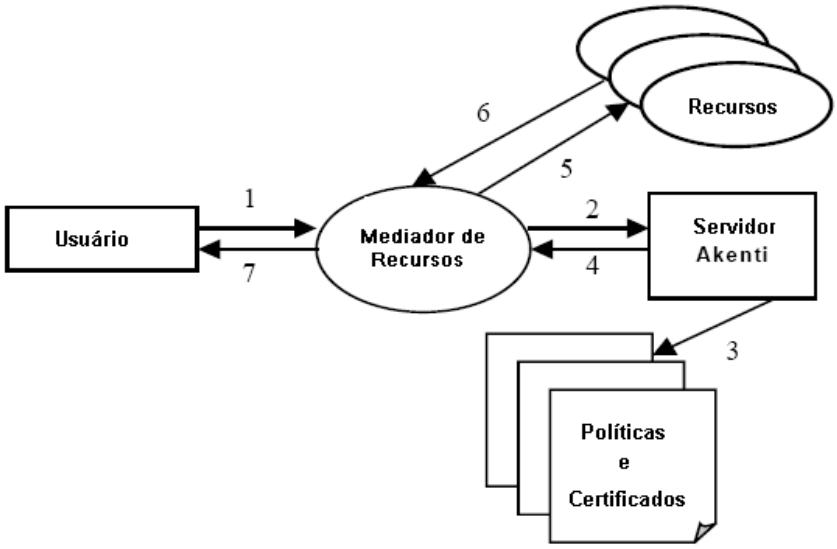

Figura 3.12: Modelo de autorização do AKENTI, adaptado de Thompson et al. (2003)

3.3 Segurança no GT4

A parte do GT4 destinada a tratar das questões de segurança dos serviços é conhecida como *Grid Security Infrastructure* (GSI). O GSI pode ser decomposto em quatro funções distintas (Welch, 2005): proteção das mensagens, autenticação, delegação e autorização. Diferentes padrões são usados para prover essas quatro funcionalidades:

- Protocolo TLS (Dierks et al., 1999) é adotado para prover segurança e confidencialidade no nível de transporte. Já os padrões WS-Security (OASIS, 2006) e WS-SecureConversation (OASIS, 2007a) atuam no nível de mensagem, utilizados em mecanismos de proteção de mensagem em combinação com SOAP (Mitra e Lafon, 2007).

- Certificados X.509 são utilizados como mecanismo para autenticação. Além disso, a autenticação pode ser feita com base no uso de identificador de usuário (*username*) e senha como forma alternativa.

- Certificados *Proxy* X.509 e o padrão WS-Trust (OASIS, 2007b) são usados para delegação de credenciais.

- Asserções (conjunto de afirmações) da linguagem *Security Assertions Markup Language* (SAML) (OASIS, 2005) são utilizadas para autorização.

Conforme mostra a Figura 3.13, as quatro funções apresentadas podem ser implementadas por meio de diferentes perfis. O primeiro perfil trata da segurança em nível de mensagem, com o uso de certificados X.509, em conformidade com o padrão WS-Security. Entretanto, esse modo de operação oferece um baixo desempenho. O segundo perfil utiliza um identificador do usuário (*username*) e senha como forma de autenticação. A operação é normalmente menos segura, mas pode ser utilizada por dispositivos móveis como *Personal Digital Assistants* (PDAs) e está em conformidade com o padrão de interoperabilidade estabelecido pelo WS-I Base Security Profile. O último perfil é implementado utilizando o protocolo *Transport Layer Security* (TLS). O GT4 usa o protocolo SSL/TLS sobre HTTP para garantir a segurança da comunicação entre o cliente e o servidor e credenciais X.509 são usadas para autenticação. Esse perfil é o garante melhor desempenho, sendo o padrão de configuração da segurança do GT4.

Os componentes do GT4 que oferecem serviços dentro do padrão de serviços *web* usam o *Simple Object Access Protocol* (SOAP) (Mitra e Lafon, 2007) como protocolo de troca de informações estruturadas para comunicação. O GT4 oferece dois mecanismos para proteção das mensagens SOAP que são transferidas entre seus diferentes componentes: segurança em nível de transporte e em nível de mensagens.

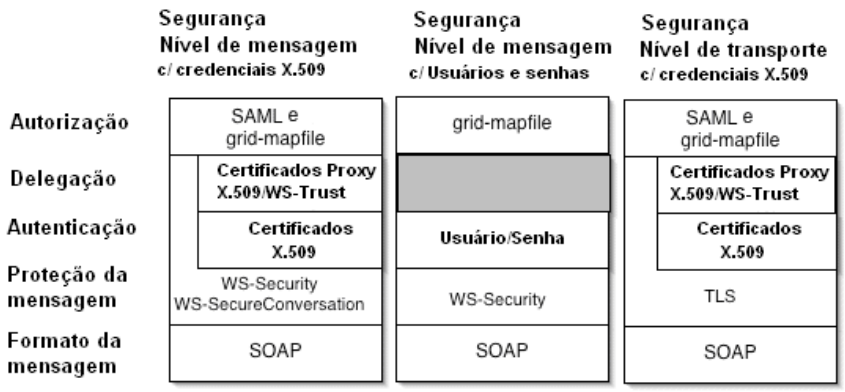

Figura 3.13: Diferentes funções do GSI, adaptado de Welch (2005)

Com relação à segurança em nível de transporte (*Transport-Level Security*), o GSI exige que as mensagens SOAP sejam transmitidas sobre uma conexão de rede protegida pelo protocolo TLS, que oferece privacidade e integridade na comunicação. Ela é o mecanismo de proteção das mensagens padrão do GT4 e é a alternativa que oferece o melhor desempenho. Ela é normalmente usada em conjunto com credenciais X.509 para permitir autenticação, mas pode ser usada sem credenciais para prover proteção das mensagens (integridade) sem autenticação, conhecido como transporte anônimo. Nesse caso, a autenticação pode ser passada para a camada de aplicação através de um controle baseado em identificador de usuário e senha encapsulado em mensagem SOAP ou mantendo a comunicação verdadeiramente, sem nenhum tipo de autenticação.

Opcionalmente, com relação à segurança em nível de mensagem (*Message-Level Security*), o GSI utiliza padrões baseados em serviços *web*, tais como: WS-Security (OASIS, 2006) e o WS-SecureConversation (OASIS, 2007a). WS-Security é um arcabouço que permite a criptografia e assinatura de mensagens sobre SOAP. WS-SecureConversation tem finalidade de gerenciar a criação de contexto de segurança e a criação de chaves que podem ser usadas nesse contexto. Esse contexto pode então ser usado para proteger mensagens subseqüentes sem que o processo de autenticação seja repetido a cada momento, reduzindo o custo de processamento. Entretanto, WS-SecureConversation é oferecido pelo GSI, somente quando utilizado com credenciais X.509 ao contrário do padrão WS-Security que permite tanto credenciais X.509 como autenticação baseada em um identificador de usuário (*username*) e senha. A combinação WS-Security com WS-SecureConversation pode

ser usada para prover mecanismos adicionais de proteção que podem ser combinados entre si, como: integridade, encriptação e não repudiação.

O GT4 suporta também outros métodos de autenticação e autorização não baseados em serviços *web*. Eles consistem de um conjunto de *Application Programming Interfaces* (APIs) e ferramentas para autenticação, autorização e gerenciamento de certificados. A API de autenticação utiliza tecnologias de ICP (Infra-estrutura de Chaves Pública) tais como certificados X.509 e TLS. Ela fornece mecanismos de delegação baseados em certificados *Proxy* X.509. O suporte para autorização vem de uma pequena coleção de APIs. A primeira delas suporta o controle de acesso baseado em credenciais do cliente, tais como encadeamentos de credenciais X.509. A segunda oferece um modelo baseado em *Access Control List* (ACL) que é uma lista de controle de acesso que mapeia entidades remotas autorizadas para nomes de usuários locais. Além dessas, há várias outras APIs de mais baixo nível e ferramentas para gerenciar, descobrir, e consultar certificados.

3.4 Portais de Grade

Portais e *portlets* são tecnologias emergentes que ganham popularidade na comunidade de desenvolvedores, pois facilitam a criação de software para usuários finais de maneira mais ágil, fácil e padronizada. O portal é uma aplicação *web* que normalmente permite fácil personalização, oferece suporte para autenticação única, agregação de conteúdo de diferentes fontes (*portlets*) e trata de todas as questões relativas à camada de apresentação de um sistema de informação (Abdelnur e Hepper, 2003). A agregação é a ação de integrar conteúdo de diferentes fontes dentro de uma página *web*. Um portal pode exibir um sofisticado recurso de personalização para prover configuração de conteúdo para seus usuários. Páginas de portal podem conter diferentes conjuntos de *portlets* que disponibilizam conteúdo adaptados para diferentes usuários (Abdelnur e Hepper, 2003). Os *portlets* são componentes *web* baseados na tecnologia Java gerenciados por um contêiner de *portlets* que processam requisições e geram um conteúdo dinâmico. Os *portlets* são usados por portais como componentes de interface para usuário, podendo ser conectados e desconectados no portal (Abdelnur e Hepper, 2003). Os *portlets* são compatíveis com um padrão conhecido como *Java Specification Request 168* (JSR-168), cujo foco é garantir a

interoperabilidade entre *portlets* e o contêiner de *portlets*. Os *portlets* desenvolvidos em conformidade com esse padrão podem ser instalados em qualquer contêiner de *portlets* compatível com a especificação JSR-168 (Abdelnur e Hepper, 2003) sem modificação do seu código fonte.

Um contêiner executa *portlets* e provê suporte aos requisitos necessários ao ambiente de execução. Um contêiner de *portlets* acondiciona *portlets* e gerencia o seu ciclo de vida. Ele também provê a persistência do armazenamento das preferências do *portlet*, e recebe requisições do portal para executar operações nos *portlets*. O contêiner de *portlets* não é responsável pela agregação do conteúdo produzido pelos *portlets*, pois isso é uma tarefa do portal. O portal e o contêiner de *portlets* podem ser construídos em conjunto como um simples componente de uma aplicação ou como dois componentes separados (Abdelnur e Hepper, 2003), conforme ilustra a Figura 3.14.

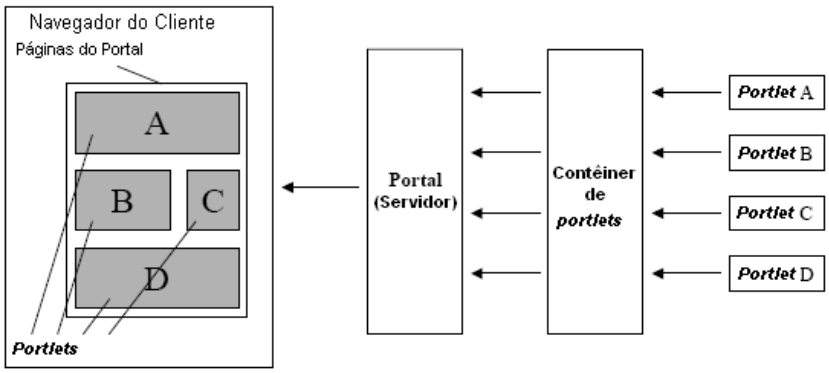

Figura 3.14: Construção de páginas do Portal, adaptado de Klaene (2004)

As grades computacionais podem ter muitos serviços que podem ser disponibilizados aos seus usuários por uma interface gráfica e um *portlet*. Essa combinação é chamada de *portlet* de grade (*grid portlet*) e capacita os *portlets* a interagirem com os serviços disponíveis em uma grade computacional. Por exemplo, um *portlet* de autenticação pode ser criado para validar a identidade dos usuários de um portal. Outro, pode ser desenvolvido para permitir que os usuários tenham acesso aos arquivos remotos de algum repositório da grade. Outro *portlet* pode permitir submissão de tarefas ao ambiente *grid* onde vários usuários podem compartilhar uma interface comum no portal para acessar os recursos *grid* (Cai et al., 2006).

A Figura 3.15 ilustra a combinação de um arcabouço de portal, contêiner de *portlets* e *portlets* de grade que juntos compõem um portal de grade. Os *portlets* de grade interagem

com os demais componentes da própria plataforma de grade computacional (como o GT4, por exemplo) ou com componentes da aplicação que executam na grade (Cai et al., 2006).

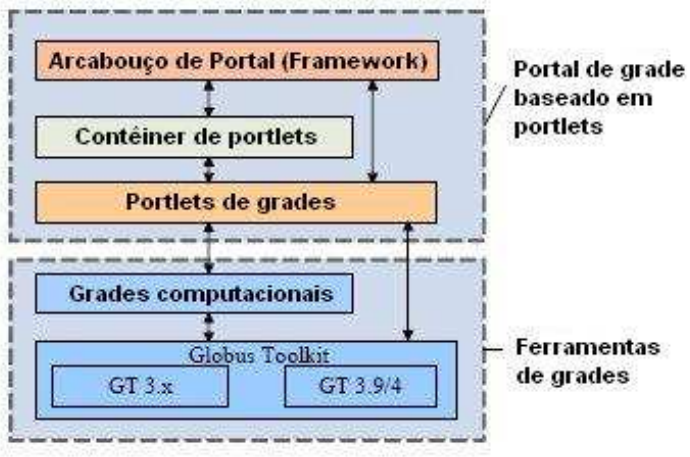

Figura 3.15: Arquitetura de um *portlet* baseado em portal de grade, adaptado de Cai et al. (2006)

Portlets de grade acessam recursos da grade computacional através do GT ou através de outras plataformas *grid* invocando seus componentes ou serviços. Os serviços de grade são introduzidos entre os serviços de mais baixo nível oferecido pelo GT e a camada de apresentação que é oferecida pelo portal de grade. Diferentes composições de *portlets* de grade oferecem aos usuários finais diferentes funcionalidades.

Os portais de grade oferecem vários benefícios. i) Oferecem um único ponto de acesso para os recursos da grade, tipicamente por meio de uma interface *web*. ii) Reduzem a exposição natural de recursos que as grades computacionais normalmente não o fazem. iii) Oferecem um mecanismo único de autenticação. iv) Permitem personalização da interface *web*. v) Permitem um acesso transparente do cliente aos recursos da grade.

É importante que a tecnologia *grid* seja avaliada considerando as necessidades de segurança, e que as melhores práticas sejam estabelecidas para executar projetos baseados em portais de grade. Mais adiante, no decorrer dessa seção, serão explicadas com mais detalhes as ferramentas Clarens, GridSphere e OGCE, que dão suporte à implementação de portais de grade.

3.4.1 A Ferramenta GridSphere

O GridSphere é um projeto *open-source* desenvolvido como parte do projeto GridLab, financiado pela Comissão Européia em 2002, e tem como principal finalidade a criação de um arcabouço para o desenvolvimento de portais para grades computacionais baseado no conceito de *portlets*. A partir da versão 2.0, ele passou a ser compatível, também com o produto WebSphere da IBM (versão 4.2). Isso garante que *portlets* desenvolvidos e executados no produto da IBM possam facilmente ser portados para o GridSphere. Já a compatibilidade com o padrão JSR-168 garante a compatibilidade com outros produtos desenvolvidos por vários líderes de mercado, que também adotaram essa especificação, tais como IBM, Sun, Oracle entre outras. GridSphere é um projeto que conquistou espaço na comunidade *grid*. Dentre os projetos que adotaram a ferramenta GridSphere se pode citar o programa de e-Science do Reino Unido, o D-Grid (Deutsche Grid-Initiative) alemão e o K*Grid (Korean National Grid Program) coreano.

Pelo fato do GridSphere possuir somente um núcleo básico de funcionalidades para desenvolvimento de portais *web*, foi preciso desenvolver o suporte adequado para acesso aos serviços de grade por meio de uma aplicação *web* chamada de *Grid Portlets*. A primeira versão dessa aplicação foi lançada em junho de 2005 e oferece um conjunto de *portlets* para gerenciamento de recursos e credenciais, submissão de tarefas e transferência de arquivos entre repositórios *grid*.

A aplicação *Grid Portlets* tem uma arquitetura em camadas conforme mostra a Figura 3.16. A camada de apresentação é responsável por oferecer recursos que facilitem a construção de interface com os usuários do portal. Ela oferece uma coleção de *portlets* bem integrados que podem ser utilizados pelo próprio portal GridSphere ou em outros portais. Esses *portlets* são construídos com a tecnologia *Java Server Pages* (JSP) baseada em componentes de interface reutilizáveis, chamados de componentes de ação (*action components*). A título de exemplo, essa camada implementa o *portlet Job Submission* que utiliza o componente *File List* para listar e pesquisar os arquivos que serão submetidos à execução no *grid*. Esse mesmo componente pode ser reutilizado por outros componentes do próprio *Grid Portlets* ou para o desenvolvimento de aplicações de terceiros, o que facilita o desenvolvimento de aplicações.

A camada da lógica de negócio é responsável por uma coleção de serviços que permite a execução de tarefas no *grid*. Esses *portlets* definem uma API para interagir com os recursos do *grid*. Finalmente, o nível mais baixo dessa camada completa essa arquitetura permitindo a ligação com uma particular infra-estrutura ou tecnologia *grid*. A título de exemplo, o serviço *File Browser* oferece métodos para listar arquivos em repositório remotos do *grid*.

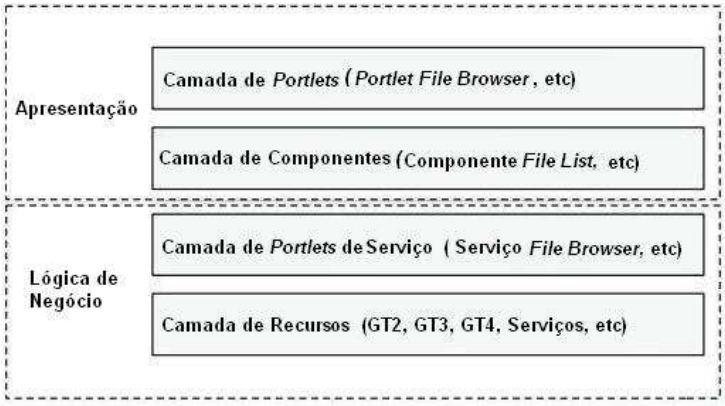

Figura 3.16: Grid Portlets tem uma arquitetura em camadas, adaptado de Novotny et al. (2006)

O núcleo básico do GridSphere oferece as principais funcionalidades de um portal como o gerenciamento de usuários, segurança (autenticação e autorização), personalização das configurações, facilidade e agilidade no desenvolvimento de aplicações. O GridSphere possui um modelo de alto nível para construção de *portlets* mais complexos, utilizando componentes reutilizáveis Java de sua biblioteca. Oferece suporte à persistência de dados através de um arcabouço para o mapeamento de objeto para o modelo relacional que facilita o acesso a vários tipos de bancos de dados, por exemplo, MySql, Oracle, Postgres, HSQL, etc. Ele também possui uma biblioteca de testes integrada, o JunitCactus que automatiza a avaliação dos serviços disponibilizados pelos *portlets*. Cada aplicação desenvolvida para o GridSphere gerencia sua própria base de dados, o que permite uma total independência da base do portal.

No quesito segurança, o GridSphere oferece um mecanismo de controle de acesso baseado em papéis. O mecanismo de autenticação é processado por módulos configurados pelo administrador do portal, que são similares ao *Pluggable Authentication Modules* (PAM) suportado pelos sistemas UNIX. Módulos específicos também podem ser desenvolvidos para atender a um determinado mecanismo de autenticação. O GridSphere suporta au-

tenticação baseada em identificador do usuário e senha, JAAS, LDAP e MyProxy (só estará disponibilizado no portal após a instalação da aplicação *Grid Portlets*). Se mais de um módulo de autenticação for configurado, ele irá consultar a todos, de acordo com uma ordem de prioridade pré-configurada (Vecchio et al., 2006). A autorização utiliza o conceito de controle de acesso baseado em papéis. Cada *portlet* só poderá ser utilizado por usuários que tenham privilégios suficiente para tal operação. Cada usuário autenticado precisa estar associado a um papel específico para obter a permissão de acesso ao *portlet*. Um usuário pode ter um ou mais papéis associado a ele. Um usuário que não passou pelo processo de autenticação pode apenas visualizar a página inicial do portal. Para registro das atividades dos usuários no portal, o GridSphere oferece um mecanismo simples de rastreabilidade das atividades dos usuários por meio da biblioteca Log4j que é amplamente utilizado pela comunidade Java.

3.4.2 A Ferramenta OGCE

O OGCE (*Open Grid Computing Environments Collaboratory*) é um outra ferramenta para construção de portais de grade. Ele foi iniciado em 2003 com financiamento do programa Middleware Initiative da NSF (National Science Foundation). Como é um projeto *open-source*, desenvolvido por um consórcio formado pelo Laboratório Argonne, Universidade de Indiana, Universidade de Michigan, NCSA (National Center for Supercomputing Applications) e o TACC (Texas Advanced Computing Center). O OGCE é um conjunto de *portlets* que pode trabalhar com diferentes contêineres de *portlets*, tais como: o GridSphere, o uPortal, o Jetspeed e outros que estejam em conformidade com o padrão JSR-168. Para uma discussão mais profunda, para efeito deste trabalho será considerado o uPortal como provedor da infra-estrutura necessária para execução dos *portlets* do projeto OGCE. Portanto, a arquitetura dessa solução de uPortal/OGCE pode ser comparada àquela utilizada pelo projeto GridSphere/*Grid Portlets* pois muitos dos seus requisitos são determinados pela especificação JSR-168.

Uma das vantagens do uPortal reside no fato de que este é o arcabouço mais amplamente utilizado pela área acadêmica devido às contribuições dadas pelas várias instituições acadêmicas em função de suas próprias necessidades (Akram et al., 2005). Ele é um ar-

cabouço bastante estável (Akram et al., 2005). Pelo fato da primeira versão do uPortal ter sido liberada antes da especificação JSR-168, foi utilizado um mecanismo não previsto no padrão, chamado de canal, onde os recursos disponíveis estão baseados em soluções personalizadas e locais. Ou seja, canais foram utilizados no lugar de *portlets*. A versão mais recente do uPortal suporta *portlets* compatíveis com a especificação JSR-168 através da utilização de adaptadores que utilizam a ferramenta *open-source* Pluto, que é uma implementação de referência do padrão JSR-168. O suporte direto para *portlets* é esperado em futuras versões. Com relação à interface de apresentação do usuário, um par de folhas de estilos XSLT é usado para converter o *layout* de um usuário, que é definido em um arquivo XML, para uma estrutura de página com o tema desejado. A persistência de dados é delegada para o Spring JDBC, que é similar ao conhecido *Java Database Connectivity* (JDBC), porém, com algumas características que permitem construir conexões de forma mais fácil, e mapear objetos Java na lógica relacional do banco de dados.

A base da arquitetura OGCE, conforme ilustra a Figura 3.17, é composta de serviços e *portlets*. O OGCE usa Java CoG Kit (Laszewski et al., 2001) como sua principal API de serviços para acessar os recursos disponibilizados pela grade. O OGCE implementa um conjunto de *portlets* de mais alto nível que interagem com outros componentes e serviços da grade. Um *portlet* denominado, por exemplo, *Grid Portal Information Repository* (GPIR) é usado para recuperar informações, como por exemplo, o *status* da carga ou o tamanho da fila de execução da grade. O gerenciador do MyProxy (*MyProxy Manager*) do OGCE permite que outros *portlets* acessem recursos *grid* por meio de credenciais recuperadas pelo *portlet* MyProxy.

Um conjunto de *portlets* fazem parte do pacote básico do OGCE que utiliza o suporte da biblioteca Java CoG Kit (Laszewski et al., 2001) para prover a camada de abstração necessária para o acesso aos serviços de grade, são eles:

- *MyProxy Manager Portlet*: É um *portlet* gerenciador de *proxy* com a finalidade de obter credenciais de servidores MyProxy.
- *File Manager Portlet*: Esse é um *portlet* gerenciador de arquivos que tem o objetivo de listar arquivos remotamente, além de enviar e receber arquivos através do serviço GridFTP.

- *Job Submission Portlet*: Esse é um *portlet* responsável pela submissão de tarefas para execução remota utilizando o serviço GRAM.

- *Information Services Portlet*: É um *portlet* que utiliza o GPIR para fazer o controle das características e informação de *status* de vários recursos computacionais globais e locais.

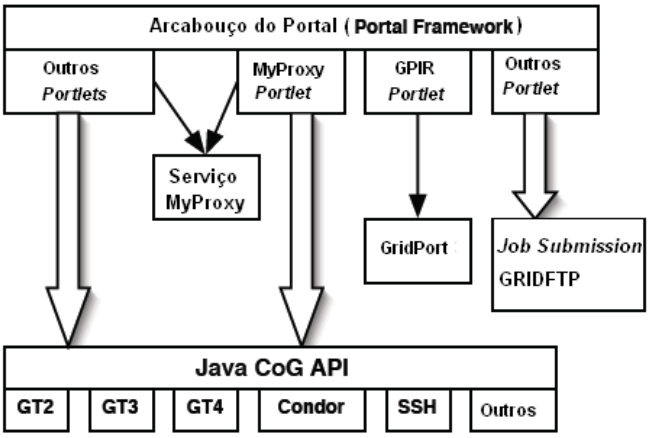

Figura 3.17: Arquitetura do OGCE, adaptado de Zhang et al. (2007)

A tecnologia usada pelo serviço de autenticação do uPortal é semelhante às tecnologias utilizadas pelo GridSphere, permitindo opções de autenticação de usuários, como por exemplo, o serviço LDAP. A configuração padrão oferece um canal (similar a um *portlet*) de autenticação do usuário que coleta a informação necessária e a redireciona para um *servlet* responsável pela autenticação (Vecchio et al., 2006).

A unidade básica de informação manipulada pelo mecanismo de autorização do uPortal é chamada de *Permission*, sendo concedida pelos proprietários do recurso (geralmente um canal) a um determinado usuário. Cada *Permission* pode especificar um intervalo durante o qual ela é válida e aplicável a um determinado recurso. Por exemplo, uma permissão pode ser concedida a uma entidade (usuários membros de grupos e grupos) para permitir a execução de uma tarefa em um canal. O sistema de autorização do uPortal trabalha com o conceito de grupos, que permite a uma entidade herdar permissões concedidas ao grupo. A ferramenta uPortal possui duas interfaces que podem ser usadas para configurar permissões para canais ou *portlets* e uma outra para grupos, denominadas respectivamente de *ChannelManager* e *GroupManager*. O uso de interfaces abstratas para representar as

classes desse mecanismo de gerenciamento sugere que futuramente poderão ser incluídos novos mecanismos de autorização, embora isso não esteja claro na atual versão (Vecchio et al., 2006).

3.4.3 A Ferramenta Clarens

O Clarens é um projeto que teve início em 2001 com o objetivo de fornecer um arcabouço escalável para desenvolvimento de aplicações distribuídas, baseadas em serviços *web*. Inicialmente, o projeto seria parte do experimento CMS (Compact Muon Solenoid Technical Proposal) do CERN. Com a adoção de serviços *web* e serviços de grade para desenvolvimento de aplicações distribuídas, Clarens tornou-se parte dos requisitos de diversos outros projetos de pesquisa. Sua implementação em linguagem Java é chamada de JClarens, enquanto que a implementação escrita em Python, é chamada de PClarens (Lingen et al., 2005). A versão JClarens é baseada em *servlets* para TOMCAT. A implementação em Java, apesar de arquitetura semelhante, não será considerada nessa discussão.

Diferentemente de outras soluções de portais, o Clarens permite que *portlets* sejam desenvolvidas em ambos os lados, tanto do cliente como do servidor, o que faz com que os clientes não estejam restritos ao uso de navegadores *web*. O servidor Clarens é implementado como uma extensão do servidor Apache, utilizando o módulo para a linguagem Python (mod_python). O servidor Apache recebe as requisições POST ou GET de um cliente e invoca o servidor PClarens baseado no cabeçalho HTTP e no conteúdo da requisição. A comunicação segura pode ser habilitada entre o cliente e o servidor Apache de forma transparente, pelo servidor PClarens. Depois que a requisição do cliente tiver sido processada, a resposta será enviada de volta ao cliente normalmente, como uma resposta POST codificada como XML-RPC ou SOAP. Esse processo é semelhante ao mecanismo de resposta de mensagens de erro do servidor *web*. A Figura 3.18 ilustra a arquitetura do projeto Clarens para a linguagem Python.

Como Clarens utiliza requisições baseadas no protocolo HTTP, não armazena o estado das conexões. Entretanto, é importante que a informação sobre as sessões dos usuários sejam armazenadas em um banco de dados. Para isso, a ferramenta utiliza o chamado

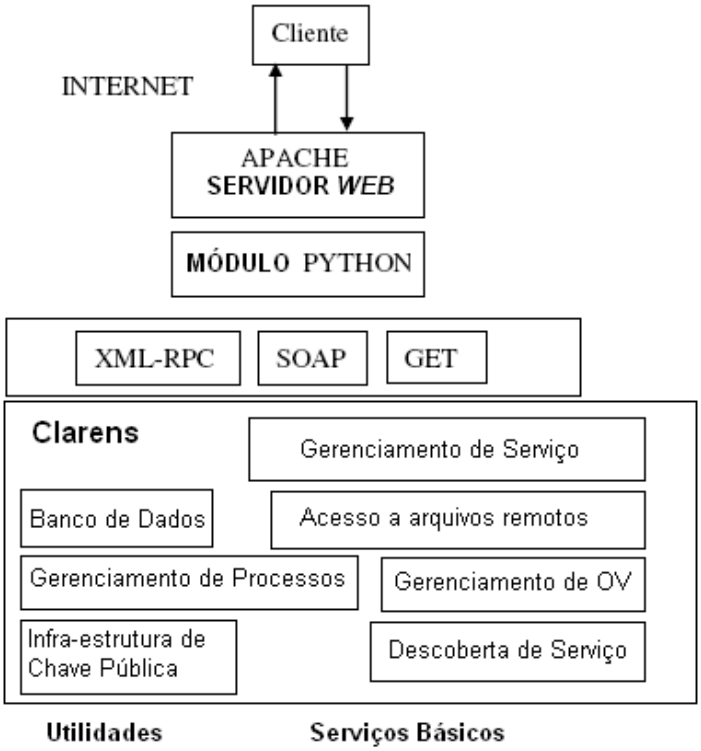

Figura 3.18: Arquitetura do Clarens, adaptado de Lingen et al. (2005)

Trivial DataBase (TDB), que suporta múltiplos pedidos de gravação que utiliza um sistema interno de controle de conexões concorrentes (Vecchio et al., 2006). O núcleo de um servidor Clarens contém um conjunto de funcionalidades básicas, tais como:

- Segurança (autenticação e autorização).
- Descoberta de serviços.
- Gerenciamento de sessão (com persistência de informação)
- Mecanismo de registro de atividades dos usuários (*logging*)
- Gerenciamento de módulos instaláveis (*plug-in*).

O Clarens implementa alguns serviços que permitem acesso aos recursos *grid*, tais como:

- *Proxy*: oferece um serviço de gerenciamento de credenciais X.509 semelhante ao MyProxy.
- Gerenciamento de arquivo: permite o acesso local e remoto a arquivos.

- Shell: serviço que permite a execução de comandos remotos.
- Registro e descoberta: permite a busca, registro, e verificação da disponibilidade dos recursos do sistema.

A segurança oferecida pelo Clarens é baseada em certificados X.509 e o protocolo SSL. O serviço de *proxy* do Clarens oferece um meio seguro de armazenar e obter certificados *proxy* X.509 para serem utilizados na invocação dos serviços de grade. O serviço *proxy* tem características semelhantes ao serviço oferecido pelo MyProxy, e permite proteger o certificado *proxy* armazenado em um repositório com uma senha que será usada no momento de sua recuperação, para executar qualquer serviço de grade.

O processo de autenticação é iniciado através da invocação de um método RPC com a informação de identificador de usuário e senha. O servidor responde com o seu certificado, juntamente com um identificador de sessão do servidor criptografado com a chave pública do usuário, e um identificador de sessão do cliente criptografado com a chave privada do servidor. Isto assegura que somente a chave privada do cliente pode descobrir o identificador da sessão do servidor e que o servidor possui a chave privada que consta no certificado. Um vez completado esse procedimento, o certificado do cliente e do servidor podem ser, também verificados por uma Autoridade Certificadora (Ali et al., 2004).

O mecanismo de autorização do Clarens é baseado em listas. Uma lista pode ser criada para definir quem tem permissão ou não de acessar um particular módulo ou método desse módulo. O controle de acesso para arquivos é similar aos módulos e permite definir permissões para arquivos ou diretórios. Os usuários no Clarens são identificados por seus *Distinguished Names* (DNs) e esse DNs pode estar associados a uma hierarquia de organizações virtuais de grupos e subgrupos. Os grupos são organizados em uma estrutura chamada de arvore ternária (Bentley e Sedgewick, 1998) para agilizar uma posterior pesquisa dessa informação (Steenberg et al., 2003). Membros de grupos de níveis mais altos são automaticamente membros de grupos que estão nas folhas abaixo desses grupos. O grupo do administrador está localizado na raiz dessa árvore. Esses grupos ou DNs de usuários são utilizados para definir uma política de controle de acesso para módulos, métodos ou arquivos. O Clarens oferece interface *web* de administração para configuração desses grupos e listas que facilita a definição e controle das permissões

dos usuários, mas pode se tornar mais confuso que o processo de edição de arquivos de configuração (Vecchio et al., 2006).

3.5 Segurança em Portais de Grade

No passado, os portais eram tipicamente sítios *web* para pesquisa e indexação de conteúdo para outros sítios distribuídos globalmente. A preocupação com a segurança era reduzida, pois geralmente todo o conteúdo de portais *web* era disponibilizado para qualquer usuário.

Entretanto, para a área de computação em grade os recursos de interesses não estão em sítios *web*, mas são recursos computacionais, bases de dados, serviços e aplicações que pertencem a instituições e indivíduos colaboradores. Portais de grade podem disponibilizar acesso a tais recursos por meio de uma interface *web*. Os portais oferecem funcionalidades básicas de interação com o usuário, além de permitir a execução remota ou local de tarefas, recuperação ou armazenamento de dados e acesso a recursos pertencentes a outros domínios administrativos. Os recursos oferecidos pelas grades, também podem envolver por exemplo, um conjunto de *clusters* de alto desempenho ou um dispositivo de armazenamento de alta capacidade contendo informação classificada, confidencial ou particular de algum domínio ou usuário.

Portanto, a questão de segurança em portais de grades ganha amplitude devido à natureza dos recursos de grades que é exposta por meio de alguma falha na segurança do portal.

No quesito da segurança da informação, os princípios básicos confidencialidade, integridade, disponibilidade e rastreabilidade aplicados aos serviços *web* podem ser aplicados integralmente na segurança dos recursos disponibilizados pelos portais de grade (Vecchio et al., 2006). Em uma grade computacional, confidencialidade e integridade são necessárias para poder transmitir dados sobre uma rede aberta como a Internet. Além disso, os recursos disponíveis são caros, poderosos e de operação delicada, como exemplo, um sofisticado microscópio eletrônico. Portanto, a utilização desses recursos deve ser feita somente por pessoas autorizadas. A disponibilidade de recursos é vital para o funciona-

mento de uma grade computacional sem o qual não será possível o compartilhamento de recursos. Finalmente, rastreabilidade é um requisito importante, pois permite registrar quais usuários executaram quais tarefas.

Os portais de grade têm a vantagem de que as grades possuem seus próprios mecanismos de segurança *in loco*, e, mesmo que um atacante ultrapasse os limites estabelecidos para a segurança do portal, ele não vai necessariamente obter acesso aos recursos da grade. Por exemplo, o simples fato de um atacante ter acesso a uma aplicação no portal que submete tarefas à grade, não é útil sem posse de uma credencial apropriada para sua identificação e autorização para invocação ao serviço adequado. Conseqüentemente, a chave para o desafio da segurança na maioria dos portais é que, em algum momento, os portais gerenciam as credenciais em nome dos clientes. Credenciais comprometidas provocam uma falha de segurança extremamente grave, pois permite ao atacante efetivamente personificar um usuário válido da grade até sua credencial ser revogada ou expirada.

Hoje em dia, a maior parte dos portais segue dois caminhos com o objetivo de permitir acesso aos serviços de grade:

1. Cada usuário do portal tem sua própria credencial. Nesse caso, é assumido que os usuários já possuem acesso aos recursos oferecidos pela infra-estrutura da grade. O papel do portal é simplesmente oferecer uma interface amigável ao recurso. Nesse cenário, um usuário autenticado no portal tem acesso a todas as aplicações do portal e o controle de acesso é gerenciado pela infra-estrutura do recurso. Entretanto, essa é uma solução que dificulta o gerenciamento e o controle de acesso devido à duplicidade de controles oferecidos por ambas as ferramentas (portal e grade). Esse problema torna-se mais evidente quando tais usuários apresentam necessidades e privilégios de acessos diferenciados na aplicação *web* e no recurso.

2. Vários usuários compartilham uma única credencial na grade. Conseqüentemente, é assumido que todos os usuários terão o mesmo nível de acesso aos recursos e a autorização para acesso aos recursos é feita na camada de aplicação do portal. Essa alternativa ignora o controle local da infra-estrutura da grade. Não é uma solução confortável do ponto de vista da segurança, pois um atacante que assuma

a identidade de qualquer usuário do portal vai possuir acesso a todos os recursos oferecidos pela grade.

A *Federal Information Security Management Act* (FISMA) (NIST, 2002) é uma lei federal dos Estados Unidos que foi promulgada em 2002 para reforçar a segurança na infra-estrutura de rede de computadores e promover as melhores práticas para sua aplicação nas áreas de governo. A seção IA-4 (*Identifier Management*) da FISMA é uma coleção representativa de diretrizes que orientam uma organização para um melhor gerenciamento da segurança dos seus usuários. A seguir, estão listados algumas das recomendações da FISMA, aplicáveis à administração da segurança de portais de grade:

1. Cada usuário deve possuir uma identificação única.
2. Cada usuário deve ter sua identidade verificada.
3. O administrador deve receber autorização de uma organização oficial apropriada para expedir a identidade de um usuário.
4. Assegurar que a identificação do usuário seja expedida para a finalidade pretendida.
5. Desabilitar a identificação do usuário depois de um intervalo de inatividade.
6. Arquivar a identidade dos usuários.

Quando cada usuário do portal possui a sua própria conta (*grid credential*) na grade, o processo de registro de um usuário no portal é uma questão menos crítica, pois um atacante não pode executar tarefas na grade somente possuindo uma conta no portal. Embora este caso ofereça um menor potencial de risco, as orientações do FISMA devem ser seguidas, independentemente da relação entre contas do portal e do *grid*. Independente dessa relação entre o portal e as credenciais *grid*, o fato é que portais de grade precisarão gerenciar contas do *grid* e é uma atitude sensata proteger essas contas. A integridade e a confidencialidade dessas contas devem ser preservadas mesmo em caso de erros ou falhas. Acessos devem ser monitorados e registrados continuamente, para identificar comportamentos suspeitos ou acessos indevidos. As credenciais armazenadas em disco devem ser protegidas do acesso de outros usuários ou aplicações.

Um dos desafios no desenvolvimento de aplicações *web*, e por extensão, a dos portais de grade, é que uma vulnerabilidade em qualquer um dos seus componentes pode resultar no comprometimento de todo o sistema. Mesmo na hipótese de que o código da aplicação esteja livre de erros passíveis de serem explorados por alguém mal intencionado, as vulnerabilidades podem existir em outros componentes, como por exemplo, o contêiner *web*. Outro desafio da segurança é a complexidade da arquitetura de muitas aplicações *web* que geralmente dificulta uma configuração apropriada do sistema. Ou seja, mesmo que um sistema esteja livre de erros que não exista nenhuma vulnerabilidade, um servidor mal configurado pelo administrador pode comprometer sua segurança.

Open Web Application Security Project (OWASP) (OWASP, 2004) é um projeto aberto à comunidade focado, na segurança de aplicações. A comunidade composta de especialistas da área de segurança de vários países compartilham seu conhecimento para produzir uma lista das dez vulnerabilidades mais críticas que afetam a segurança de aplicações *web* e conseqüentemente afetam os portais de grades. As vulnerabilidades mais comuns foram reagrupadas para facilitar nossa discussão:

- Parâmetros não avaliados: As entradas contidas na requisição de uma chamada *web* não foram adequadamente verificadas antes de serem executadas pelo portal. Um atacante pode utilizar essa vulnerabilidade para obter acesso indevido ou perpetrar algum ato indevido. Esse tipo de vulnerabilidade represetam alguns tipos específicos que não serão objeto de discussão para este trabalho, mas que fazem parte da lista da OWASP.

- Falha no controle de acesso: Esse problema ocorre quando o mecanismo de controle de acesso opera de forma inconsistente ou incorretamente e permite o acesso indesejável aos recursos.

- Falha no gerenciamento da sessão e da autenticação: Problemas de autenticação ocorrem pela utilização de mecanismos considerados fracos, ou seja, que são facilmente quebrados por um atacante. Um exemplo disso constitui, o uso de frases com respostas obvias para recuperação de senhas. Já a falta de proteção adequada da informação sobre a sessão dos usuários, permitirá a um atacante forjar a identidade do usuário e obter acesso aos recursos permitidos para o usuário legítimo.

- Falha no tratamento adequado dos erros: Esse tipo de vulnerabilidade ocorre quando mensagens de erros mostradas para os usuários de alguma forma revelam detalhes da estrutura do código ou da operação. Um atacante pode obter conhecimento do sistema a ser atacado por meio do conteúdo dessas mensagens.

- Armazenamento impróprio de informação: Ao armazenar informações sensíveis, como por exemplos senhas ou informação sobre o usuário, sem o tratamento apropriado, (criptografia ou mecanismo de controle de acesso) quando associado a outras vulnerabilidades, facilita a atividade de um atacante para forjar a identidade ou acesso de um usuário.

- Negação de Serviço: Essa vulnerabilidade permite que um atacante bloqueie o acesso aos usuários legítimos por meio da sobrecarga ou interrupção do serviço. Por exemplo, um atacante poderia gerar uma quantidade de tentativas de conexão que exceda a capacidade de armazenamento de eventos da unidade onde se encontra instalado o serviço e provocar a interrupção do serviço para usuários legítimos.

- Insegura configuração: A falta de correção ou atualização dos produtos utilizados, configuração padrão insegura definida para o ambiente de produção, falta de conhecimento sobre o produto ou uma incompleta configuração aplicada são alguns dos vários motivos que provocam esse tipo de vulnerabilidade. Geralmente essa é uma falha provocada por um problema humano, entretanto um produto de software que fosse facilmente compreendido, simples de instalar e configurar seria de grande ajuda para definir uma configuração segura.

3.6 Conclusão

Nos últimos anos, tecnologias de grades computacionais têm sido empregadas em diversos projetos da área científica e comercial. No campo da engenharia civil, as grades computacionais são utilizadas para dar suporte ao trabalho colaborativo de uma comunidade de especialistas (Pearlman et al., 2004). Conforme foi visto nesse capítulo, padrões como OGSA e WSRF têm se tornado cada vez mais importantes. O GT4, a implementação de referência da arquitetura OGSA, está entre as ferramentas mais utilizadas

atualmente. Com o crescimento das soluções de portais para acesso aos serviços oferecidos por uma grade, os chamados portais de grade, as questões relativas à segurança passaram a ser melhor avaliadas. A questão de autenticação para portais de grade parece estar consolidada em torno do uso de certificados X.509 e na adoção de certificados *proxy* como solução para autenticação única (*Single Sign-On*) e para delegação a serviços *web*. Também, já é um procedimento muito usual, os desenvolvedores de soluções para portais adotarem o MyProxy como repositório de credenciais.

A questão da autorização em soluções de portais de grade ainda é tratada separadamente entre a grade e o portal. Muitos projetos criam portais que adotam soluções bastante simplistas para o controle de acesso, que consiste em mapear todas as identidades do portal para uma única identidade credencial na grade, o que demonstra a falta de soluções melhores e de fácil acesso, atualmente. Isso gera alguns inconvenientes administrativos e de segurança, como por exemplo, se para resolver um incidente de segurança envolvendo um acesso do portal aos serviços da grade for necessário desabilitar a conta compartilhada adotada por essa solução, todos os usuários do portal serão afetados. Todavia, criar uma credencial na grade para cada usuário do portal e gerenciar individualmente as permissões associadas a cada identidade também não é adequado em se tratando de comunidades com grandes quantidades de usuários.

Nenhum dos três projetos avaliados (GridSphere, OGCE e Clarens) apresentou uma solução de gerenciamento integrada de autorização entre o portal e a grade. A falta de integração e complexidade para configurar as permissões dos usuários e dos recursos pode introduzir vulnerabilidades ou inconsistências no sistema (Vecchio et al., 2006).

Nenhum dos três portais analisados foi particularmente forte no quesito rastreabilidade. Isso é especialmente verdadeiro para a solução Clarens que mostrou a ausência de um mecanismo configurável de registro de *log*. O pacote Log4J, utilizado pelas outras duas soluções, certamente oferece condições para viabilizar um controle mais apurado e permitir um mecanismo de auditagem em um formato consistente. No entanto, da forma que está sendo oferecido pelos três portais, seria muito difícil analisar e detectar problemas relacionados a falhas de segurança e muito menos, corrigi-las(Vecchio et al., 2006).

Todos os três produtos deveriam ser distribuídos com o controle de acesso desabilitado até o administrador explicitamente configurar a política e mecanismos de controle condizente com a organização. Entretanto, suas configurações padrão contrariam as boas práticas de segurança sugeridas pela FISMA (NIST, 2002).

O próximo capítulo mostrará os detalhes de uma proposta para resolver alguns desses problemas de segurança levantados na solução de portais de grades.

Capítulo 4

Proposta IAGP

Grades computacionais normalmente envolvem o uso de um grande número de recursos computacionais, instituições e indivíduos que se associam, formando redes colaborativas que compartilham e coordenam o acesso a tais recursos (Foster e Kesselman, 2004). Uma dificuldade enfrentada atualmente pelos desenvolvedores de aplicações para grades computacionais consiste na disponibilidade de soluções que permita controlar o acesso a uma quantidade significativa de usuários e recursos geograficamente dispersos, envolvendo domínios administrativos diferentes (Kostopoulos et al., 2007). Tais demandas tornam-se ainda mais evidentes quando tais usuários apresentam necessidades e privilégios de acessos diferenciados.

Uma tendência observada nos últimos anos é adoção de padrões arquitetônicos, como por exemplo, a arquitetura OGSA para grades computacionais. Uma outra tendência observada é a utilização de soluções de portais para desenvolvimento de aplicações. Os portais simplificam o acesso aos recursos e serviços por parte de usuários e aceleram o desenvolvimento de novas aplicações. Especificações relativas à padronização no desenvolvimento de portais têm sido criadas para facilitar a portabilidade de aplicações entre os diversos fabricantes dessas soluções. Um exemplo é a especificação JSR-168 (Microsystems, 2003).

Alguns desenvolvedores de aplicações para portais fazem uso de infra-estrutura de grades para criar um ambiente distribuído com suporte ao desenvolvimento colaborativo de recursos e obter uma maior escalabilidade da solução proposta. Um exemplo é a

aplicação de *e-learning* desenvolvida pelo grupo de pesquisa dos autores (Rosatelli et al., 2006) em conjunto com os pesquisadores do LabVirt (Nunes, 2002) que estabelece um ciclo de produção de objetos de aprendizagem (*learning objects*) que envolve professores e alunos de escolas de ensino médio, educadores e alunos do ensino superior. Uma rede colaborativa para um ambiente desse tipo envolve centenas de instituições de ensino, alunos, programadores, especialistas em conteúdo e catalogação para compartilhamento de materiais para aprendizagem eletrônica e produção de objetos de aprendizagem. Por essa razão, é importante do ponto de vista da segurança oferecer um mecanismo que permita o controle de acesso a esses recursos. Dentro desse cenário, um aspecto muito importante seria promover a colaboração de um grande número de usuários, com diferentes papéis e diversos níveis de privilégios de acesso aos recursos, localizado em diversas instituições que compõem a rede colaborativa formada. A proposta apresentada no presente trabalho surgiu em função dessa necessidade.

Este capítulo propõe a arquitetura **I**ntegrated **A**uthorization for **G**rid **P**ortal (IAGP) para desenvolvimento de aplicações para grades computacionais que permite integrar os mecanismos de autenticação e autorização previstos no modelo OGSA com as facilidades de gerenciamento de identidades e o controle de acesso para as aplicações oferecidas pelo portal.

Pela proposta IAGP, um usuário pode obter acesso ao portal por meio de um navegador *web* padrão a partir de um computador, celular ou PDA, mas precisa ter sua identidade verificada pelo portal para obter acesso às informações ou serviços não disponíveis para usuários não registrados (anônimo). O portal verifica a identidade do usuário por meio de um repositório de credenciais. Após o usuário ter sido reconhecido como um usuário cadastrado no portal, os recursos permitidos para o seu perfil, configurado pelo administrador são disponibilizados para seu uso. A qualquer momento que o usuário enviar requisição para um serviço na grade protegido pelo IAGP, a invocação é interceptada e a permissão concedida pelo administrador é verificada por meio de um repositório de regras de autorização. Se o usuário possui a autorização necessária para acesso ao serviço solicitado, o IAGP libera o acesso ao serviço, caso contrário, é negada a execução do serviço. A Figura 4.1 ilustra o funcionamento da abordagem proposta.

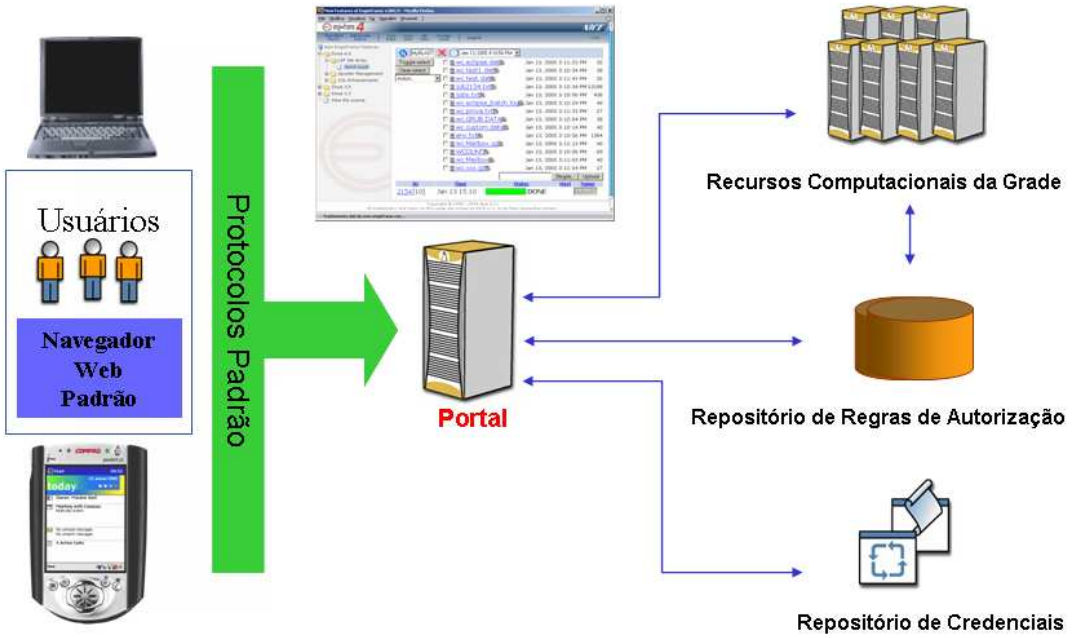

Figura 4.1: Funcionamento básico do IAGP

O desenvolvimento do IAGP foi iniciado com a especificação dos requisitos funcionais relativos à necessidade de segurança de uma aplicação de *e-learning* baseados em portal que utiliza recursos colaborativos oferecidos por uma infra-estrutura de grades. A partir desses requisitos foi definida a arquitetura que levou o desenvolvimento de um protótipo e permitiu a avaliação dessa proposta.

4.1 Requisitos Funcionais

Dentro do cenário apresentado na seção anterior, foram definidos o escopo e os requisitos funcionais dessa proposta. Um processo de pesquisa bibliográfica sobre as principais propostas de mecanismos de autorização dedicados a grades e portais complementaram o levantamento dos principais requisitos funcionais necessários para o desenvolvimento dessa proposta, tais como:

- Uso de RBAC: O modelo de controle de acesso baseado em papéis é uma alternativa interessante para definir permissões sobre os recursos ou serviços disponibilizados pelo portal e pela grade. RBAC é um modelo já amadurecido que oferece esse mecanismo de autorização baseado em papéis. A grande vantagem do modelo RBAC é a sua independência de políticas de acesso ao recurso. O administrador dos recursos

pode definir suas permissões aos recursos e associá-las a papéis sem precisar conhecer previamente a quais usuários serão concedidas essas permissões. Permissões baseadas em papéis oferecem uma alternativa mais escalável e com custo menor de administração. Por exemplo, num portal com 1.000.000 de usuários e 10 papéis, o administrador precisa definir permissões somente para os 10 papéis e não para cada um dos usuários. Geralmente, o conjunto de usuários pode ser significativamente maior que o conjunto de papéis. Além disso, o conjunto de usuários sofre alterações mais freqüentemente do que o conjunto de papéis. Um conjunto mais estável e reduzido de papéis associados às permissões permite reduzir a complexidade do gerenciamento e aumentar a escalabilidade da solução. São pontos muito interessantes para serem aplicados em grades computacionais.

- Gerenciamento Integrado: A maioria das soluções de portais de grades trata de forma separada o gerenciamento dos usuários e das permissões do lado do cliente (portal) e do provedor de serviço (grade). A solução mais comum é utilizar uma única credencial na grade compartilhada com todos os usuários do portal e administrar a autorização somente na camada da aplicação do portal. Nessa alternativa, qualquer usuário registrado no portal passa a ter idênticos direitos na grade. Isso não atende a um princípio clássico de segurança conhecido como *need-to-know* (Lento et al., 2006). Esse princípio diz que um usuário deve ter acesso somente à informação ou recurso que ele precisa utilizar nas suas atividades. Na prática, esse princípio limita o dano que um usuário legítimo pode fazer no sistema, permitindo o acesso somente aos recursos mínimos necessários para executar suas tarefas. O gerenciamento integrado permite que as mesmas permissões que o usuário tem no portal sejam estendidas para o ambiente da grade, evitando erros na configuração das permissões aos serviços e facilita o gerenciamento dos usuários, papéis e serviços por meio de uma única ferramenta. Portanto, é desejável ter uma ferramenta integrada de gerenciamento de autorização que permita estender as permissões do portal para a camada da infra-estrutura de serviços de grades computacionais.

- Certificados X.509: O principal conceito usado para identificação de usuários e assegurar a segurança em grades são os certificados digitais. Certificados são usados

para autenticação e para criação de uma versão temporária do mesmo, chamada de certificado *proxy*. Certificados *proxy* permitem a autenticação única e a delegação de tarefas na grade. Entretanto, aplicações *web* não oferecem mecanismo de delegação de serviços como fazem as grades. Uma alternativa para resolver esse problema e oferecer um único mecanismo de autenticação para o portal e a grade é utilizar um repositório *on-line* de credenciais conhecido como MyProxy. Soluções de portais para grades computacionais como por exemplo, OGCE e *Grid Portlets* já fazem uso desse serviço. O uso de certificados padrão X.509 por meio do serviço MyProxy é uma alternativa desejável para desenvolver o mecanismo de autenticação para portais baseados em grades.

- Rastreabilidade: O registro de atividades dos usuários é um item importante na segurança de qualquer sistema. As soluções de portais de grade apresentadas anteriormente, OGCE, *Grid Portlets* e Clarens delegam para o portal ou criam seu próprio mecanismo de contabilidade das atividades dos usuários do portal. Entretanto, grades fazem o registro de atividades de modo independente do portal. Isso dificulta a correlação das atividades do portal e da grade pelo administrador. Uma solução conveniente para esse problema seria ter um único mecanismo de rastreabilidade para que o administrador possa ter a mesma visão integrada das atividades dos usuários no portal e na grade.

- JSR-168 (Microsystems, 2003): Para facilitar a portabilidade entre as diversas soluções de portais é necessário adotar a especificação JSR-168 no desenvolvimento de novas aplicações de portais. Por exemplo, os portais GridSphere (GridSphere, 2005) e uPortal (Akram et al., 2005) estão em conformidade com esse padrão. Teoricamente, uma aplicação desenvolvida no GridSphere pode ser executada também no uPortal. Portanto, esse é um requisito importante na definição de arcabouço de desenvolvimento de aplicações para portais.

4.2 Visão Geral

Integrated **A**uthorization for **G**rid **P**ortal (IAGP) é um arquitetura voltada a atender às necessidades de segurança dos desenvolvedores de aplicações para portais baseadas em grades computacionais. O IAGP combina as facilidades de gerenciamento e segurança oferecidas pelo portal com o arcabouço de autorização disponibilizado pelo GT4 para oferecer uma alternativa de segurança pronta para o uso, sem necessidade de alterações no código das aplicações instaladas no portal ou na grade.

Essa proposta oferece três funcionalidades básicas que atendem os requisitos comuns de segurança de aplicações de portais :

- Autenticação: Adiciona uma nova funcionalidade no portal que permite a identificação única dos usuários por meio do serviço MyProxy (repositório de credenciais) que é uma ferramenta já inclusa no GT4.

- Autorização: Estende o controle de acesso baseado em papéis do portal para permitir a proteção também de serviços registrados no GT4 e obter um único mecanismo para definição das permissões dos usuários e dos serviços.

- Contabilidade (*Accounting*): Adiciona uma novo recurso no portal que permite uma visão integrada das atividades dos usuários em relação ao processo de autenticação no portal e a autorização obtida com a chamada de serviços. Facilita o monitoramento e a identificação de possíveis falhas na segurança.

A Figura 4.2 ilustra a interação entre os elementos básicos do IAGP que é composto dos seguintes componentes:

- IAGP-Login: É o componente que adiciona uma nova funcionalidade à solução do portal que foi tomada como referência para permitir que o usuário defina o seu papel no momento que ele tiver sua identidade verificada pelo mecanismo de autenticação no portal.

- IAGP-AuthzRules: Esse componente implementa uma interface para um banco de dados que modela as entidades e permite a autorização aos serviços baseada no papel do usuário definido no processo de identificação do usuário no portal.

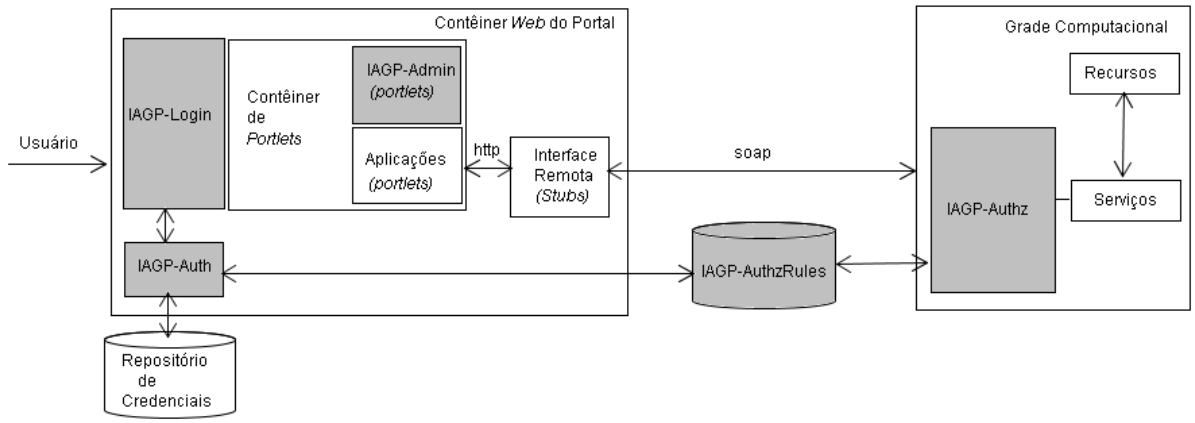

Figura 4.2: Visão geral da proposta IAGP

- IAGP-Auth: Ele é o componente responsável pela validação da identidade dos usuários registrados no portal. Esse componente faz interface com o repositório de credenciais, implementando as funções de armazenamento e recuperação de credenciais de usuário no repositório, além da criação de credenciais temporárias (certificado *proxy*) ao iniciar a sessão no portal.

- IAGP-Admin: Esse é um componente que oferece dois *portlets* para facilitar a administração do portal. O primeiro *portlet* permite configurar as permissões aos serviços e o segundo permite visualizar as atividades dos usuários.

- IAGP-Authz: Esse é o componente que trata a autorização do lado do provedor de serviços. Ele captura as mensagens de invocação aos serviços e verifica se o usuário possui permissão para executar o serviço requisitado baseado no papel escolhido pelo usuário durante o processo de autenticação.

Um usuário que deseje acesso a informações que exigem a verificação de sua identidade pelo portal deve, primeiramente, possuir um certificado X.509 armazenado no repositório de credenciais. Será considerado nessa proposta que todo usuário do portal possui um certificado X.509 armazenado no repositório. Sem passar pelo processo de autenticação, o usuário só tem acesso às informações e/ou serviços públicos disponíveis no portal.

Para ser autenticado, um usuário deve fornecer algumas informações básicas, tais como um identificador (*username*), uma senha e o papel assumido na sessão corrente. O componente IAGP-Login estende as funcionalidades do portal e permite que o usuário possa ter mais de um papel, mas somente um papel pode estar ativo por sessão corrente. O controle

da identificação dos usuários é gerenciada pelo componente IAGP-Auth que fornece uma interface para o serviço MyProxy. No final do processo de autenticação ocorre o registro do *status* da atividade do usuário e a atualização dos atributos utilizados para definir quais os serviços autorizados para o papel corrente do usuário. O armazenamento dessas informações é gerenciado pelo componente IAGP-AuthzRules. O usuário autenticado tem permissão de acesso aos *portlets* e aos serviços protegidos pelo IAGP que seu papel ativo permite acesso.

Quando o usuário interage com qualquer aplicação do portal que utilize serviços da grade, uma extensão do mecanismo de autorização do portal é invocada automaticamente pelo arcabouço de autorização da grade. O componente IAGP-Authz é responsável pela proteção dos serviços da grade baseada nas permissões do papel dos usuários e dos serviços. Caso o usuário tenha permissão de acesso requerida para acesso ao serviço, o componente do IAGP-Authz delega a execução do serviço para a credencial do usuário recuperada pelo componente IAGP-Auth. Do contrário, o acesso é negado. No final do processo de autorização é executado o componente IAGP-AuthzRules para registrar o *status* do evento de autorização do serviço chamado pelo usuário.

O gerenciamento das permissões aos serviços na grade é feito por meio do componente IAGP-Admin que permite definir os papéis requeridos para acesso aos serviços disponíveis na grade. Esse componente possui uma segunda funcionalidade que permite ao administrador do portal verificar a situação do evento relacionado a autenticação no portal e a autorização nos serviços da grade.

Desse modo, a proposta IAGP agrega as facilidades de gerenciamento do portal com um melhor controle de permissões aos serviços instalados na grade. As permissões dos usuários são configuradas baseadas em um conjunto de papéis que é significativamente menor do que a quantidade de usuários registrados em um sistema de grade. Também, não é preciso compartilhar uma única conta na grade para atender a todos os usuários do portal, pois cada usuário acessa os serviços e aplicações com sua própria credencial. A definição de permissões e papéis por meio de ferramentas de administração do portal facilita a configuração e proporciona um controle integrado menos sujeito a erros para o ambiente do portal e da grade. Por exemplo, o *log* integrado oferecido pelo IAGP facilita

a avaliação do administrador no caso de algum incidente de segurança envolvendo o acesso de um usuário do portal. O IAGP oferece aos desenvolvedores uma solução de autorização para portais de grade, sem haver necessidade de alteração do código fonte da sua aplicação tanto do lado do cliente como do lado do provedor de serviço.

4.3 Especificação dos Casos de Uso

Três tipos de usuários são reconhecidos pelo gerenciamento de usuários do portal e correspondem aos papéis básicos da ferramenta do portal: administrador (*admin*), usuário comum(*user*) e anônimo (*guest*). A Figura 4.3 ilustra uma visão geral dos casos de uso previstos e os atores envolvidos. Um usuário do portal que possua o papel de administrador será o responsável pelo gerenciamento dos usuários, criação de *layouts*, associação de papéis a todos os usuários registrados e mapeamento dos papéis para usuários da grade. Já um usuário comum registrado no portal terá permissão para executar qualquer aplicação que esteja associado ao seu papel, além das informações públicas disponíveis. Opcionalmente, alguns serviços poderão ser disponibilizados para o papel anônimo, caso seja necessário. Outros papéis podem ser criados somente pelo administrador para atender às necessidades de gerenciamento das aplicações.

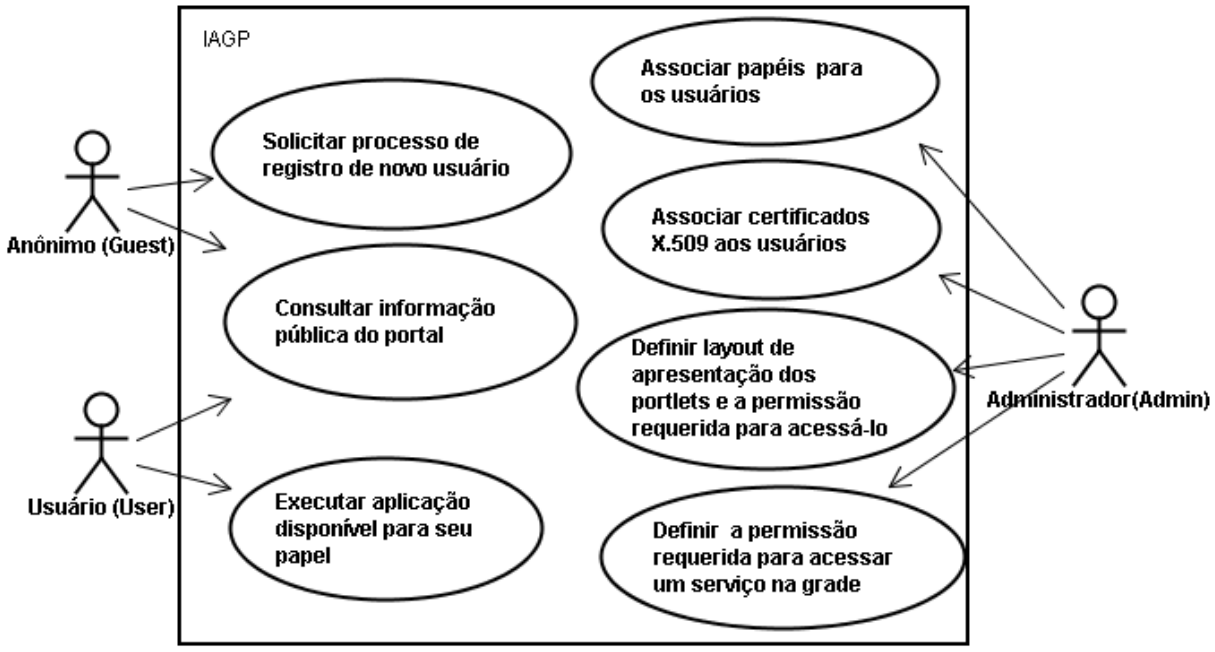

Figura 4.3: Casos de uso dos atores do IAGP

As principais funcionalidades do pacote IAGP podem ser resumidas na descrição textual por meio dos seguintes cenários:

- Cenário da Autenticação.
- Cenário da Autorização.
- Cenário da Administração.

Cenário de Autenticação

Nível do objetivo: Autenticar usuário.

Ator principal: Usuário registrado no portal.

Pré-condição: O usuário foi previamente registrado no portal, o seu certificado foi armazenado no repositório de credenciais X.509 e papéis foram atribuídos ao usuário pelo administrador do portal. **Cenário Principal de Sucesso:**

1. O usuário navega pela página principal do portal e seleciona o formulário de autenticação dos usuários. Para ser reconhecido pelo portal, o usuário fornece seu identificador, sua senha e o seu papel ativo que configura sua sessão no portal.
2. Antes de validar a identidade do usuário, uma rotina de atualização dos papéis definidos no portal é executada e os usuários com tempo de acesso expirados são removidos do seu cadastro do IAGP.
3. O portal consulta seu cadastro para verificar se o identificador do usuário está registrado no portal.
4. O portal verifica se o papel ativo selecionado faz parte de seu conjunto de papéis.
5. O portal confirma se a senha do usuário é a mesma que protege o certificado do usuário que está armazenado no repositório de credenciais X.509.
6. O portal autentica o usuário.
7. O portal registra o acesso do usuário na base de dados do portal para posterior consulta.

8. O portal retorna ao usuário uma página com um *layout* previamente definido pelo administrador, que exibe apenas os *portlets* que são permitidos para o papel ativo definido no item 4.

A Figura 4.4 ilustra o diagrama de seqüência que mostra o processo de autenticação do IAGP.

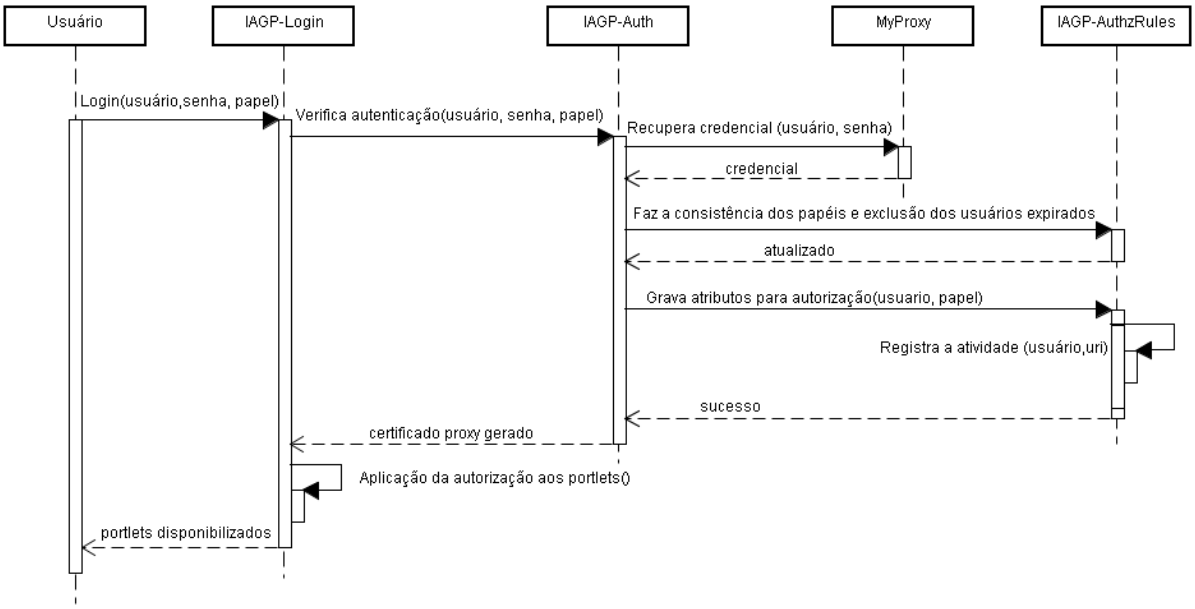

Figura 4.4: Autenticação do IAGP

Cenário Alternativo:

Passo 1a: O usuário prefere navegar no portal sem ser identificado tendo acesso somente ao conteúdo associado ao papel anônimo (*guest*).

Passo 3a: O identificador do usuário não foi encontrado no banco de dados do portal.

1. O identificador do usuário preenchido no formulário de autenticação não é reconhecido pelo portal e o acesso do usuário é recusado pelo portal.

Passo 4a: O papel selecionado no formulário de autenticação não está associado para esse usuário.

1. O portal recusa a autenticação do usuário pois o papel selecionado não está associado ao mesmo.

Passo 5a: A senha do usuário está incorreta.

- A senha informada no formulário de autenticação está incorreta e a entrada do usuário é recusada pelo portal.

Cenário de Autorização

Nível do objetivo: Autorizar a execução dos *portlets* e serviços na grade.

Ator principal: Usuário autenticado.

Pré-condição: O usuário deve estar autenticado.

Cenário Principal de Sucesso:

1. Os *portlets* que o usuário autenticado têm acesso são carregados pelo portal e disponibilizados para o usuário. Para cada papel do usuário existe um *layout* que define a página de apresentação do usuário e o papel requerido para carregamento dos *portlets*. O usuário executa uma operação por meio dos *portlets* que invoca a um serviço na grade.

2. A requisição é interceptada por um elemento do arcabouço de autorização da grade (PEP) do Globus Toolkit 4. Esse mecanismo executa dois componentes do mecanismo de autorização: PIP e PDP.

3. O PIP fornece a lista de serviços autorizados para o papel definido na sessão do usuário no portal por meio do identificador do usuário.

4. O PDP fornece a decisão da autorização no acesso ao serviço a partir da lista de serviços autorizados fornecida pelo PIP e da URI do serviço chamado. Caso, o serviço chamado esteja nessa lista, o PDP responde ao PEP que o serviço foi autorizado.

5. O portal registra o acesso do usuário na base de dados do portal para posterior consulta.

6. Uma vez autorizado, o PEP aplica a decisão tomada pelo PDP e retorna a resposta para o usuário do portal.

A Figura 4.5 ilustra o diagrama de seqüência que mostra o processo de autorização do IAGP.

Cenário Alternativo:

Passo 4a: Falha na execução do serviço IAGP.

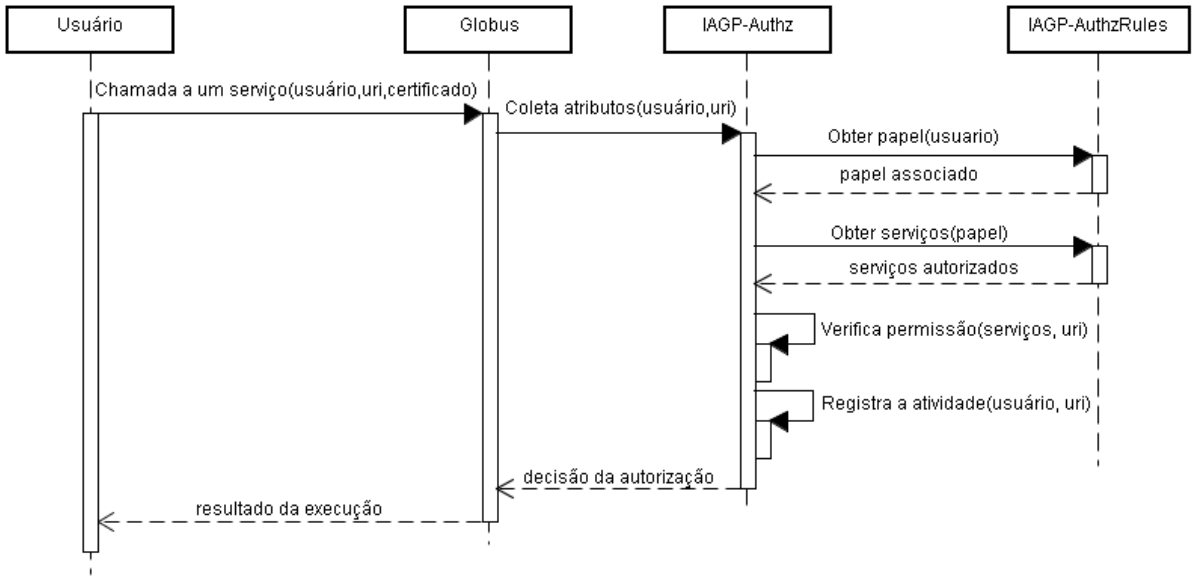

Figura 4.5: Autorização do IAGP

1. O serviço IAGP-Authz não está ativo na grade ou falhou na inicialização do serviço.

Passo 5a: Falha na comunicação com o serviço IAGP.

1. O componente IAGP-AuthzRules falhou ou perdeu a comunicação com o banco de dados.

Passo 6a: Falha na execução do serviço de grade chamado pelo usuário.

1. O *grid* sinaliza um erro no registro de eventos do sistema.

Cenário da Administração

Nível do objetivo: Fazer o registro de um novo usuário.

Ator principal: O usuário mapeado no papel de administrador.

Pré-condição: O certificado do usuário deve estar armazenado no repositório de credenciais, o usuário deve estar autenticado e as permissões requeridas para acessar os *portlets* e serviços foram definidas pelo administrador.

Cenário Principal de Sucesso:

1. Os *portlets* que o administrador autenticado tem acesso são carregados pelo portal associado ao seu papel ativo.

2. O administrador interage com o portal e requisita o registro de um novo usuário no portal.

3. Os atributos cadastrais sobre o novo usuário, previamente verificados por um processo não digital, são inseridos pelo administrador.

4. O administrador associa um ou mais papéis para esse novo usuário e define uma senha provisória para acesso ao portal.

5. O administrador finaliza o processo de registro e o portal armazena as informações sobre o usuário no seu banco de dados.

Cenário Alternativo:

Passo 5a: Falha no registro do usuário.

1. O portal sinaliza um erro no seu mecanismo de registro de atividades do sistema e é necessário o suporte do administrador do sistema.

4.4 Aspectos da Implementação

Essa seção detalha o desenvolvimento do protótipo **I**ntegrated **A**uthorization for **G**rid **P**ortal (IAGP) que foi implementado para validar suas funcionalidades.

A proposta IAGP foi construída baseada em componentes que se integram ao portal Gridsphere e da infra-estrutura de serviços do GT4 para oferecer segurança e um controle de acesso diferenciado para os serviços e aplicações.

Os componentes IAGP-Login, IAGP-Auth e IAGP-Admin são executados exclusivamente pelo GridSphere. IAGP-AuthzRules é uma interface para acesso ao banco de dados MySQL. O IAGP-Authz é um componente instalado na ferramenta GT4. O conjunto de componentes do IAGP foi desenvolvido em Java 1.5 de acordo com o padrão de cada ambiente. A implementação do IAGP está relacionada com os componentes distribuídos em três grupos: portal, grade e banco de dados.

4.4.1 Componentes de Portal

O componente IAGP-Login adiciona duas novas funcionalidades ao GridSphere. A primeira permite ao usuário definir o seu papel durante o processo de autenticação e a segunda executa o carregamento da aplicação *web* (*portlets*) de acordo com o papel escolhido por ele.

Para adicionar essas funcionalidades ao portal, o IAGP-Login herda a classe LoginPorlet do portal e modifica o método *login()* que possui duas assinaturas diferentes para operações distintas (sobrecarga do método). A primeira operação coleta os dados do formulário incluindo o papel do usuário e a segunda redireciona a página do portal de acordo o *layout* definido pelo papel do usuário. Para dar suporte às alterações no método login() foram criados dois novos métodos para a classe LoginPortlet. O método reWriteURL(uri, *layout*) reescreve a URL que carrega o *layout* com a definição dos *portlets* associados ao papel do usuário. E o método chooseRole(req, *user*) que armazena e disponibiliza o papel escolhido pelo usuário para o mecanismo de autenticação do portal ou para outros *portlets* que estejam sendo executados pelo portal.

O GridSphere define a apresentação e carregamento dos *portlets* somente para duas classes distintas de usuários: os não autenticados e os autenticados. Para cada uma dessas classes existe um arquivo XML associado à apresentação dos *portlets* permitidos ao seu papel, respectivamente os arquivos guest.xml e loggedin.xml. Esse arquivo XML é conhecido como *layout*. Entretanto, com a adição dessas novas funcionalidades propostas pelo componente IAGP-Login é permitido associar um arquivo de *layout* para cada novo papel do usuário. Para facilitar o desenvolvimento da presente proposta, será adotado que cada papel definido no portal tem um arquivo XML de *layout* previamente configurado pelo administrador. Uma vez criado esse arquivo, o portal oferece sua própria ferramenta de gerenciamento para alterar a configuração da apresentação dos *portlets* e definir os papéis requeridos pelos usuários para executar os *portlets*.

Para que os usuários do GridSphere pudessem se autenticar da mesma forma que os usuários do GT4, foi adicionado o componente IAGP-Auth que armazena e recupera credenciais por meio do serviço MyProxy incluso no GT4. O componente IAGP-Auth é composto pelos seguintes componentes:

- MyProxyAuthModule é o responsável pela adição de um novo mecanismo de autenticação ao portal. Sua habilitação permite a validação dos atributos passados pelo usuário no momento do login por meio da interface oferecida pelo componente MyProxyResource. Ele foi desenvolvido de acordo com um modelo adotado pelo projeto GridSphere que oferece suporte ao desenvolvimento de novos módulos, como por exemplo, o LDAP ou *Java Authentication and Authorization Service* (JAAS). O modelo adotado pelo GridSphere para adição de novos mecanismos de autenticação no portal é similar ao recurso do *Pluggable Authentication Modules* (PAM) utilizado no sistema Unix no qual cada módulo pode ser enfileirado, ativado e associado a uma prioridade para cada um deles. Isso torna possível a um usuário autenticar-se no módulo padrão do GridSphere (banco de dados local), se alguns dos outros módulos com uma prioridade mais alta falhar na primeira vez que ele for carregado pelo portal.

- MyProxyResource é uma interface para a comunicação com o serviço MyProxy. Caso a identificação seja positiva é gerado um certificado *proxy* para posterior invocação a serviços no GT4. Do contrário, uma exceção é lançada. Oferece métodos básicos de armazenamento, remoção e recuperação de credenciais. A comunicação com o MyProxy é realizada com o suporte da ferramenta *Commodity Grid* (CoG) Kit (Laszewski, 2005).

Para permitir o gerenciamento das permissões e as atividades dos usuários foram criadas duas aplicações para o portal. A primeira permite ao administrador definir as permissões aos serviços da grade e a segunda permite verificar de forma integrada os registros de atividades dos usuários. IAGP-Admin é o componente que disponibiliza esse novo recurso ao portal GridSphere.

Para a consistência das regras de autorização e a exclusão dos usuários com tempo de acesso expirados, no final de cada processo de autenticação, é delegada essa tarefa ao componente IAGP-AuthzRules que é detalhado a seguir.

4.4.2 Componentes de Banco de Dados

Para permitir que os atributos que compõem a política de controle de acesso baseada em papéis pudessem ser compartilhados entre o GridSphere e o Globus foi definido o componente IAGP-AuthzRules. Para evitar introdução de erros ou incompatibilidades com o projeto original do GridSphere foi criada uma nova estrutura que é a implementação do modelo de controle de acesso baseado em papéis (RBAC) para os usuários que tiveram sua identidade confirmada pelo processo de *login*. O componente IAGP-AuthzRules é também o responsável pelo armazenamento e recuperação das permissões necessárias para acessar cada serviço protegido pelo IAGP. Para o desenvolvimento desse componente foi adotado o padrão de projetos *Data Access Object* (DAO) que facilitou seu uso compartilhado por outros componentes do IAGP, além de diminuir a dependência do gerenciador de banco de dados.

Foi definido um modelo Entidade-Relacionamento que contém as regras da política de autorização aos serviços. A Figura 4.6 ilustra o modelo de Entidade-Relacionamento adotado pela proposta IAGP.

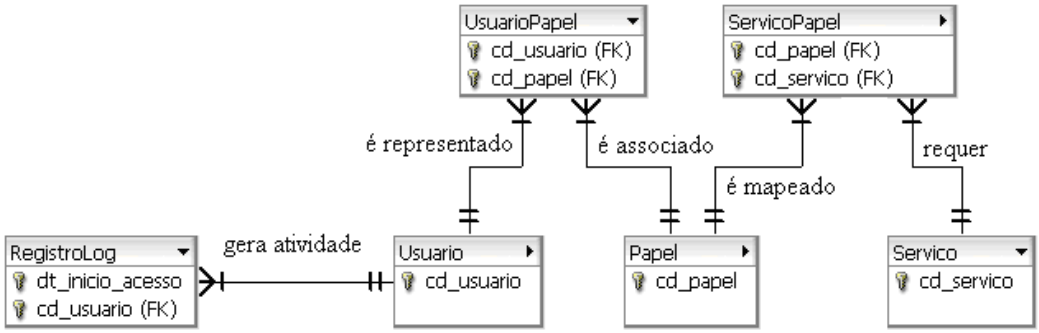

Figura 4.6: Modelo de Dados pelo componente IAGP-AuthzRules

Cada entidade do modelo conceitual está associada a um componente do IAGP-AuthzRules que permite a comunicação com o gerenciador do banco de dados MySQL (versão 5.0). IAGP-AuthzRules é utilizado pelos componentes IAGP-Auth, IAGP-Admin e IAGP-Authz para atualizar informações que permitem o acesso aos serviços e registro de atividades dos usuários. IAGP-AuthzRules é composto pelos componentes: Roles, Users, Services, UserRoles, ServiceRoles e LogRegisters. A interface com o banco de dados é

feita pelo componente DaoResourceImpl que realiza a comunicação entre os componentes do modelo adotado e a instância do MySQL.

A entidade Papel é representada pelo componente Roles que armazena os papéis dos usuários ativos no portal. A entidade Servico é representada pelo componente Services e armazena as URI dos serviços registrados no GT4. A entidade ServicoPapel é representada pelo componente ServiceRoles e armazena a associação serviço-papel. A entidade Usuario é representada pelo componente Users e contém a identificação dos usuários que possuem uma sessão aberta no portal. A entidade UsuarioPapel associa o usuário com o papel escolhido durante o processo de autenticação e é representada pelo componente UserRoles. A entidade RegistroLog é a responsável pelo armazenamento das informações relativas ao processo de autenticação e autorização da proposta e está representada pelo componente LogRegisters. Por uma questão de implementação foi retirado do modelo físico o relacionamento do componente LogRegisters. Isso evita a perda de informação das atividades dos usuários em virtude das regras de integridade do banco de dados quando houvesse, por exemplo, uma exclusão de usuário em cascata.

Para manter a consistência das informações coletadas pelos usuários com sessão aberta no portal, o componente DaoResourceImpl oferece alguns métodos que mantém atualizado o seu conjunto de papéis com a informação disponível no portal por meio do recurso de banco de dados conhecido como visão (*VIEW*). Esse recurso tem algumas restrições, mas permite que o IAGP-AuthzRules armazene uma cópia do conjunto de papéis sincronizada com o banco de dados do portal. A cada autenticação do usuário o conjunto de papéis disponibilizado por meio dessa visão atualiza o componente Roles do IAGP-AuthzRules e, assim, permite que as regras de integridades entre os componentes sejam mantidas.

4.4.3 Componentes de Grade

Cada vez que uma invocação de serviço é enviada ao contêiner do Globus, o processo de autorização é executado. Anteriormente, o usuário já teve sua identidade validada no portal e obteve uma credencial para invocação do serviço por meio do serviço MyProxy.

Para cada invocação a serviços, o contêiner envia a identificação do serviço e a credencial do usuário para o componente IAGP-Authz que requisita ao componente IagpPip o

armazenamento dos serviços autorizados para o papel do usuário. Em seguida, o IagpPdp recupera os serviços autorizados e verifica se serviço chamado faz parte do conjunto de serviços autorizados armazenados pelo IagpPip. Em seguida, repassa a decisão ao arcabouço de autorização do GT4 para aplicação da decisão tomada. Os componentes IagpPip e IagpPdp são uma implementação da interface PIP e PDP respectivamente do arcabouço de autorização do GT4.

Para habilitar a proteção de serviços por meio do IAGP-Authz, um descritor de segurança deve ser definido no arquivo conhecido como *WS Deployment Descriptor* (WSDD) que informa ao contêiner como o serviço deve ser publicado, conforme o exemplo ilustrado pela Figura 4.7.

```
<?xml version="1.0" encoding="UTF-8"?>
...
<service name="Acumulador" provider="Handler"
<parameter name="securityDescriptor"
...
<value="@config.dir@acumulador-security-descriptor.xml"/>
...
<service name/>
```

Figura 4.7: Fragmento do arquivo WSDD

Esse descritor faz parte da arquitetura do arcabouço de autorização da grade e aponta para um arquivo xml que contém a seqüência de classes que devem ser invocadas para verificar a permissão sobre o serviço chamado pelo usuário. O arquivo "acumulador-security-descriptor.xml" contém a seqüência de classes que representam os componentes interceptores (IagpPip e IagpPdp) de uma chamada de autorização a serviço, ilustrada pela Figura 4.8.

```
<securityConfigxmlns="http://www.globus.org">
...
<authz value="ascope:br.unisantos.globus.IagpPip
mscope:br.unisantos.globus.IagpPdp"/>
...
<securityConfigxmlns/>
```

Figura 4.8: Descritor de Segurança

A Figura 4.9 ilustra o modelo de autorização do GT4 adotado pela proposta IAGP.

Foi observado que é possível aplicar essa proposta para acessar serviços protegidos pelo IAGP por meio de uma aplicação tradicional, sem fazer uso do portal. Nesse modelo

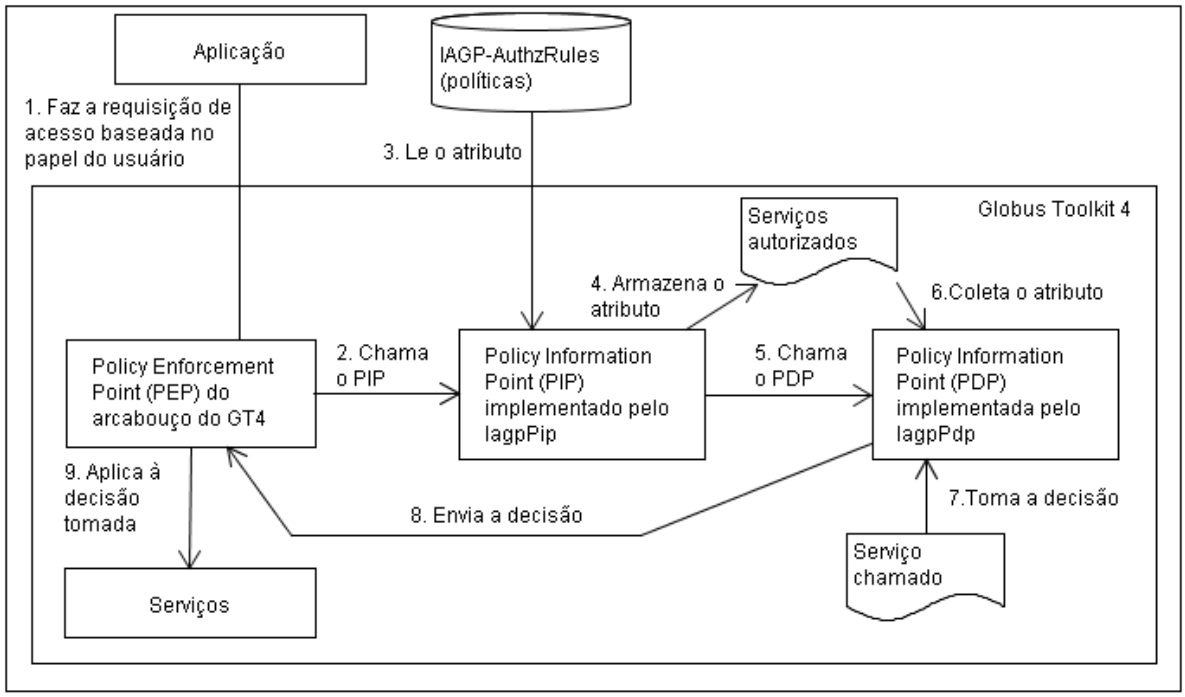

Figura 4.9: Modelo de Autorização do GT4

tradicional de acesso aos serviços no GT4, um usuário se autentica diretamente no servidor de aplicação, interage com o serviço registrado no GT4 que virtualiza um recurso por meio de uma aplicação em C, Python ou Java. Ao ser autenticado o usuário recebe um certificado *proxy* que permite fazer uma invocação ao serviço. O mecanismo padrão para autorização dos usuários aos serviços disponibilizados pelo GT4 é baseado em uma lista de controle de acesso. A autorização para acesso a qualquer serviço é concedida pela simples existência do usuário nessa lista. Nesse caso, o usuário autenticado pode invocar um serviço no GT4 por meio de um simples comando de linha. Mas, a decisão de autorização pode ser direcionada para o componente IAGP-Authz se a proteção do serviço estiver habilitada por meio de um descritor de segurança do GT4 comentado anteriormente. Nesse caso, uma solução muito simples para confirmar a viabilidade de invocar um serviço protegido pelo IAGP, sem fazer uso de uma aplicação de portal, seria autenticar o usuário por meio de um navegador não gráfico na linha de comando executado dentro do próprio servidor do portal.

Quando o usuário se autenticar por meio dessa aplicação na linha de comando, o componente IAGP-Auth é executado e, conseqüentemente, os atributos que definem as permissões aos serviços são atualizadas. Se o usuário possui permissão registrada no

componente IAGP-AuthzRules então ele pode acessar os serviços diretamente por meio dos comandos de linha tradicionais disponibilizados pela ferramenta GT4. A autorização é verificada pelo componente IAGP-Authz da mesma maneira que ocorre quanto a invocação do serviço é feita por meio de uma aplicação no portal.

O componente do arcabouço de autorização do GT4 executa a seqüência PIP/PDP contida no descritor de segurança para obter a decisão de autorização a ser aplicada no acesso ao serviço. Uma aplicação cliente poderia ser facilmente desenvolvida em Java ou em Python utilizando recursos da ferramenta *Commodity Grid (CoG) Kits* (Laszewski, 2005). Essa aplicação só precisaria executar o processo de autenticação da forma que é feita pelo componente IAGP-Auth.

4.5 Testes Aplicados

Este seção mostra com a execução de alguns casos de teste caixa preta (Myers, 1979), a viabilidade do sistema proposto. Para avaliar a proposta IAGP, bem como verificar suas limitações e sua capacidade de atender às necessidades de segurança de aplicações para portais de grade foram realizados testes funcionais. Nessa técnica de testes o protótipo é tratado como uma caixa cujo conteúdo é desconhecido, e só é possível visualizar o seu lado externo, ou seja, os dados de entrada fornecidos pelo usuário e as respostas produzidas como saída pelo mecanismo do portal e da grade.

Para a realização dos testes de autenticação e autorização foi utilizada uma aplicação de apoio à produção de objetos de aprendizagem (Stanzani, 2008). Trata-se de uma aplicação de *e-learning* na qual os usuários utilizam uma solução de portal para ter acesso aos serviços de um conjunto de repositórios de aprendizagem multi-institucionais e geograficamente distribuídos, que dá suporte colaborativo a grupos de produção de objetos de aprendizagem. Nesse sentido, um ponto fundamental para que a aplicação funcione corretamente é a existência de um mecanismo de identificação dos usuários e autorização para o acesso aos recursos e serviços dos repositórios. A aplicação de *e-learning* é composta por um conjunto de serviços e de *portlets*. Os serviços estão implementados em um contêiner GT4 e os *portlets* estão instalados no portal GridSphere. A aplicação necessita

de seis papéis para controlar o acesso aos serviços que podem ser utilizados por até mil usuários.

Na avaliação do protótipo será considerado que cada usuário teve seu certificado criado e adicionado no serviço MyProxy. Dentro da infra-estrutura oferecida pelo portal, os usuários e os papéis foram registrados por meio das ferramentas administrativas existentes no portal. Também, as mesmas ferramentas fazem as associações entre usuários e papéis. Todos os serviços a serem protegidos pelo IAGP foram cadastrados e suas permissões foram configuradas pelo administrador por meio da aplicação IAGP-ADMIN. Os serviços a serem protegidos pelo IAGP dentro do ambiente da infra-estrutura da grade, tiveram seus descritores de segurança configurados e os serviços foram reinstalados no contêiner (sem necessidade de compilação dos serviços).

Cada usuário que acessa o portal deve fornecer um identificador, senha e o papel que corresponde às entradas do caso de testes. O valor da saída esperada depende do papel do usuário e da permissão configurada para o serviço pelo administrador. A ordem de execução dessas entradas não altera o resultado final. A senha de cada usuário registrada no portal será utilizada pelo componente IAGP-Auth para validar a senha que protege o certificado armazenado no serviço MyProxy. Somente um usuário autenticado pode ser autorizado a qualquer atividade aos recursos do portal ou na grade. Existem duas principais relações baseadas no modelo RBAC definidas pelo administrador do portal. A primeira associa usuário ao seu conjunto de papéis conforme ilustra a Tabela 4.1 e a segunda (Tabela 4.2), define os papéis requeridos para permitir acesso aos serviços disponibilizados pela grade. Só foram incluídas na Tabela 4.1 os novos papéis que foram adicionados pelo administrador do portal e que não faziam parte da configuração padrão existente no GridSphere.

Tabela 4.1: Papéis permitidos aos usuários.

Usuário	**Papel**
mario	pesquisador, desenvolvedor, catalogador, revisor
maria	pesquisador, revisor
jose	desenvolvedor
luis	catalogador
helena	revisor, pesquisador
antonio	revisor, aluno
pedro	professor, revisor

Tabela 4.2: Papéis requeridos para acesso aos serviços.

Serviço	Papel
revisão	revisor
pesquisa	pesquisador
catalogação	catalogador
publicação	publicador

Por meio do critério Particionamento em Classes de Equivalência, o domínio de entrada foi dividido em classes válida e inválida (Copeland, 2003). Em seguida, de acordo com a metodologia adotada, seleciona-se o menor número possível de casos de teste, considerando-se que a escolha de um elemento de cada uma dessas classes seria suficiente para representá-las. A Tabela 4.3 mostra as Classes de Equivalência para a aplicação IAGP.

Tabela 4.3: Classes de Equivalência para aplicação IAGP.

Restrições de Entrada	Classes Válidas	Classes Inválidas
O usuário está registrado no portal?	Sim	Não
A senha do usuário está com o valor correto?	Sim	Não
O papel escolhido faz parte de seu conjunto de papéis?	Sim	Não
O papel escolhido pelo usuário permite acesso ao serviço?	Sim	Não

A partir das classes de equivalência, o seguinte conjunto de casos de teste foi elaborado conforme ilustra a Tabela 4.4. Esse conjunto contém o menor número possível de casos de teste que cubram todas as classes válidas e um caso de teste para cada classe inválida.

Tabela 4.4: Casos de Testes para todas as classes.

Usuário	Senha	Papel	Serviço	Classe
antonio	correta	revisor	revisão	válido
anônimo	vazio	anônimo	vazio	inválido
maria	incorreta	pesquisador	pesquisa	inválido
luis	correta	publicador	catalogação	inválido
helena	correta	pesquisador	revisão	inválido

Com a aplicação dos casos de testes no cenário formado pelas necessidades de uma aplicação de *e-learning* detalhado no começo dessa seção, a saída gerada pela proposta IAGP foi igual à esperada. Para completar, foi avaliada a possibilidade de proteção dos serviços da grade, sem que o usuário tenha acesso direto ao um portal. Os mesmos casos de testes e usuários com seu respectivo certificado foram usados para validar a alternativa

sem o uso direto de um portal. Foram utilizadas as ferramentas inclusas no GT4 e um cliente java foi desenvolvido para fazer a comunicação e a invocação ao serviço. O resultado por meio de um cliente *java* foi o mesmo da avaliação com uma aplicação de portal. A diferença é que o desenvolvedor do cliente java precisa fazer mais alguns passos que são transparentes para o desenvolvimento de uma aplicação de portal, por exemplo, a lógica da autenticação do usuário.

4.6 Avaliação

Neste capítulo é discutido como os requisitos definidos na seção 4.1 foram apoiados pelo desenvolvimento da proposta IAGP, e como os problemas levantados na seção 3.5 foram resolvidos.

Em relação à lista de requisitos, os principais componentes foram desenvolvidos adotando o padrão JSR-168 (Microsystems, 2003) para os componentes do lado da aplicação e o padrão X.812 (ISO/IEC 10181-3, 1995) adotado pelo arcabouço de segurança da grade para a construção do mecanismo de autorização integrado. Padrões agilizam o desenvolvimento e facilitam sua portabilidade.

O gerenciamento de usuários é um problema vital que deve ser solucionado para construção de portais de grade (Zhu et al., 2007). A maioria das propostas tratam os usuários *web* e de grades computacionais separadamente (Zhu et al., 2007). Para evitar os problemas decorrentes da duplicidade dos mecanismos de autenticação, autorização e auditoria, foi pesquisada a solução mais adequada de cada ferramenta (GridSphere e GT4) para identificação dos usuários e definição de permissão para os recursos. As recomendações de segurança da FISMA (NIST, 2002), também foram seguidas na presente proposta de integração dos mecanismos de segurança do portal e da grade.

Na proposta do IAGP, cada usuário possui sua própria credencial para acesso nas aplicações no portal e nos serviços disponibilizados pela grade. Um certificado digital assinado por uma entidade confiável (Autoridade Certificadora) garante que o usuário foi identificado. O certificado utilizado pelo IAGP para o usuário não sofreu qualquer alteração na sua estrutura e pode ser utilizado por qualquer outro serviço do GT4, diferentemente de

algumas propostas de autorização para grades, como por exemplo, CAS (Pearlman et al., 2002).

A autorização para acesso ao recurso somente é concedida após o usuário ter sua credencial validada por uma Autoridade Certificadora apropriada. A utilização do modelo RBAC reduz o custo de gerenciamento de permissões, e garante que o usuário acesse somente aquilo que foi estabelecido para o seu papel. Para aumentar ainda mais a segurança, a ação de uma credencial também é expirada após um limite de tempo (12h).

Conforme comentado na conclusão do capitulo 3, da forma que está sendo oferecido o mecanismo de rastreabilidade (*log*) dos três portais, seria muito difícil analisar e detectar problemas relacionados a falhas de segurança, e muito menos, corrigi-los (Vecchio et al., 2006). Entretanto, o IAGP oferece um sistema de rastreabilidade integrado que permite o armazenamento das identidades que fizeram uso das aplicações e dos serviços oferecidos pela solução de portal de grade. Nenhum dos três portais discutidos na seção 3.4 se preocuparam em gerar *log* com a finalidade de auditoria (Vecchio et al., 2006). O IAGP gera registros de *log* dos usuários separadamente das atividades do contêiner e das outras aplicações do portal o que facilita a administração, segurança e auditoria.

A complexidade da arquitetura de muitas aplicações *web* é um outro desafio da segurança, que mesmo sendo desenvolvidas livres de vulnerabilidades, pode levar a erros de configuração que comprometem a segurança do sistema (Vecchio et al., 2006). Entretanto, a simplicidade da arquitetura oferecida pelo IAGP facilita a configuração adequada e segura de todos seus componentes, sem que haja necessidade de alteração do código da aplicação no portal ou de qualquer serviço na grade.

Os portais de grades foram construídos para melhorar o acesso dos usuários aos recursos oferecidos pela grade. Entretanto, tais usuários normalmente pertencem a diferentes domínios de gerenciamento. Um cenário ideal do ponto de vista do usuário seria a integração desses dois arcabouços para unificação do gerenciamento de usuários (Zhu et al., 2007). O IAGP facilita o gerenciamento dos usuários, de papéis e de serviços, pois oferece ferramentas que simplificam sua configuração. O acesso do usuário pode ser bloqueado por meio de uma simples interface *web* que remove seus privilégios no sistema. Mesmo que um atacante utilize o certificado *proxy* gerado pelo processo de autenticação e de alguma

maneira consiga acessar algum serviço não protegido na grade, ele estará limitado pelo tempo de expiração do contido no seu certificado.

4.7 Conclusão

O portal GridSphere oferece ferramentas de gerenciamento de usuários e papéis que facilitam a administração de aplicações. Entretanto, um usuário logado poderia executar quaisquer ações permitidas a qualquer um dos seus papéis associados. O IAGP adiciona duas melhorias ao GridSphere para esse problema. A primeira permite ao usuário associar apenas um papel ativo por sessão. A segunda melhora a capacidade de personalização da interface que passou a exigir apenas os *portlets* associados ao papel que está ativo na sessão.

Originalmente, o GT4 provê mecanismos de segurança baseado em certificados para identificação de usuários e autorização baseada em uma lista de controle de acesso armazenada no *grid-mapfile*. Porém, não oferece ferramentas de administração. Além disso, fica a cargo do desenvolvedor implementar seus próprios mecanismos de autorização. O IAGP estende o mecanismo de autorização do GridSphere, permitindo a administração de usuários e papéis não só para o portal, mas também para a grade. As bases de dados geradas pelo portal são utilizados também pelos mecanismos de autorização da grade (PDP e PIP). A utilização das mesmas bases pelos dois mecanismos (portal e grade) proporciona total coerência entre as políticas de autorização aplicadas no portal e nos recursos (*in loco*).

A proposta IAGP mostrou que é possível melhorar as condições de autorização e controle de acesso aos recursos e serviços disponibilizados em uma grade computacional. A proteção oferecida pelo IAGP é transparente para a aplicação ou serviço. Os padrões utilizados (OGSA e JSR-168) para desenvolvimento nessa proposta permitem a portabilidade da solução de outros fabricantes. A integração das facilidades de gerenciamento de usuários e papéis do portal com o mecanismo de autorização da grade, que passa a ser oferecida com as ferramentas implementadas nessa proposta facilita a implementação de aplicações para o GT4, oferecendo uma solução simples e completa. A solução proposta permite também que sejam utilizadas aplicações cujos clientes não são executados

no portal. Nesse caso, o controle de acesso será exercido pelo mesmo mecanismo utilizado na solução baseada em portal. Além disso, a integração do mecanismo de autenticação incluso no GT4 oferece a capacidade de delegação para serviços *web* e autenticação única.

Capítulo 5

Conclusões

Grades computacionais têm sido adotadas para promover o compartilhamento, agregação e coordenação de grandes quantidades de recursos geograficamente distribuídos e multi-institucionais.

Nesse contexto, uma das dificuldades enfrentadas atualmente pelos desenvolvedores de aplicações para grades computacionais consiste na carência de soluções que permitam controlar o acesso de um grande número de usuários em ambientes dotados de quantidades significativas de recursos computacionais e serviços (Kostopoulos et al., 2007). Essa carência fica ainda mais evidente quando os usuários apresentam necessidades e privilégios de acesso diferenciados.

Uma tendência observada nos últimos anos é a adoção de padrões arquitetônicos para grades computacionais, como por exemplo o padrão OGSA (Foster e Kesselman, 2004). Outra tendência é a crescente utilização de portais para o desenvolvimento de aplicações. As soluções de portais implementadas pelas ferramentas GridSphere, uPortal e Clarens ignoram a questão da duplicidade de mecanismos de segurança entre o portal e a grade (Vecchio et al., 2006). As soluções atuais muitas vezes impõem algum tipo de compromisso, transferindo para a camada da aplicação (portal) a responsabilidade de controlar o acesso aos recursos, por exemplo no momento em que compartilham uma única credencial para acesso a todos os serviços oferecidos pelo provedor de recursos (grades computacionais).

A necessidade de uma solução simples que integre os mecanismos de segurança do portal e do provedor de serviço foi o principal fator que motivou a proposta do IAGP. O

IAGP é um arcabouço para desenvolvimento de aplicações seguras baseadas em portais. Com pouco esforço, é possível utilizá-lo para implementar um portal de acesso a serviços OGSA. Na solução proposta, cada usuário pode visualizar somente os *portlets*, bem como acessar somente os serviços da grade que forem permitidos, de acordo com o papel que estiver ativo durante a sua sessão no portal.

A proposta IAGP atende às principais recomendações sobre a identificação de usuários da FISMA (NIST, 2002). Cada usuário possui uma única identidade, que é verificada com base na entidade que assinou seu certificado digital. O IAGP oferece um mecanismo de *log* independente da estrutura do portal ou da grade, o que ajuda o administrador em um processo de auditoria, ou na ocorrência de alguma falha de segurança. Um usuário autenticado tem seu acesso condicionado a um conjunto de regras que são as mesmas aplicadas no serviços ou aplicações (*portlets*). Portanto, o IAGP trata o problema da duplicidade de controles de forma inteligente, ao estender o mecanismo de autorização e as ferramentas de gerenciamento oferecidas pelo portal para proporcionar um controle de acesso efetivo aos serviços da grade.

Entretanto, O IAGP mostrou deficiência na definição de políticas mais complexas, por exemplo, não permite atribuir direitos ou restrições para indivíduos, pois o elemento básico da política de controle de acesso é o papel.

Como trabalhos futuros, pretende-se avaliar a viabilidade e utilidade da arquitetura proposta em outras soluções de autorização para grade conhecidas como AKENTI (Thompson et al., 2003), o PERMIS (Chadwick e Otenko, 2003) e o VOMS (Alfieri et al., 2003).

Referências Bibliográficas

Abdelnur, A.; Hepper, S. JSR 168: Portlet Specification, version 1.0. Disponível em: `http://jcp.org/en/jsr/detail?id=168` . Acesso em: 30/08/2007., 2003.

Adams, C.; Lloyd, S. *Understanding Public-key Infrastructure: Concepts, Standards, and Deployment Considerations*. Macmillan Technical Publishing, 2002.

Akram, A. Working with WS-Resource properties, Part 1: Manipulating resource properties. Disponível em: `http://www.ibm.com/developerworks/edu/gr-dw-gr-wsrf1-i.html`. Acesso em: 10/01/2008., 2005.

Akram, A.; Chohan, D.; Wang, X. D.; Yang, X.; Allan, R. A Service Oriented Architecture for Portals Using Portlets. UK e-Science, AHM2005, Nottingham, UK, 2005.

Alfieri, R.; Cecchini, R.; Ciaschini, V.; dell'Agnello, L.; Frohner, Á.; Gianoli, A.; Lörentey, K.; Spataro, F. VOMS, an Authorization System for Virtual Organizations. In: *European Across Grids Conference*, 2003, p. 33–40.

Alfieri, R.; Cecchini, R.; Ciaschini, V.; dell'Agnello, L.; Frohner, A.; Lőrentey, K.; Spataro, F. From gridmap-file to VOMS: managing authorization in a Grid environment. *Future Generation Comp. Syst.*, v. 21, n. 4, p. 549–558, 2005.

Ali, A.; Anjum, A.; Azim, T.; Thomas, M.; Steenberg, C.; Newman, H.; Bunn, J.; Haider, R.; Rehman, W. JClarens: A Java Based Interactive Physics Analysis Environment for Data Intensive Applications. In: *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, Washington, DC, USA: IEEE Computer Society, 2004, p. 716–723.

Alliance, G. Projects with Globus Alliance Participants. Disponível em: `http://www.globus.org/alliance/projects.php` . Acesso em: 20/10/2007., 2007a.

Alliance, P. G. Globus: Security Descriptors. Disponível em: `http://www.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html`. Acesso em: 20/08/2007., 2006.

Alliance, P. G. The Globus Alliance. Disponível em: `http://www.globus.org/`. Acesso em: 01/12/2007., 2007b.

Baker; Buyya; Laforenza Grids and Grid technologies for wide-area distributed computing. *Software: Practice and Experience*, v. 32, n. 15, p. 1437–1466, 2002.

Basney, J.; Humphrey, M.; Welch, V. The MyProxy online credential repository: Research Articles. *Softw. Pract. Exper.*, v. 35, n. 9, p. 801–816, 2005.

Bentley, J.; Sedgewick, B. Ternary Search Trees. Dr. Dobb's Journal, disponível em: http://www.ddj.com/windows/184410528. Acesso em: 23/01/2008., 1998.

Burnett, S.; Paine, S.; Burnett, S.; Paine, S. *RSA Securitys Official Guide to Cryptography*. Berkeley, CA, USA: Osborne/McGraw-Hill, 2001.

Cai, Y.; Cao, J.; Li, M.; Chen, L. Portlet-based Portal Design for Grid Systems. In: *GCCW '06: Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops*, Washington, DC, USA: IEEE Computer Society, 2006, p. 571–575.

Chadwick, D. W.; Otenko, A. The PERMIS X.509 Role based Privilege Management Infrastructure. *Future Gener. Comput. Syst.*, v. 19, n. 2, p. 277–289, 2003.

Chakrabarti, A. *Grid Computing Security*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

Chervenak, A.; Schuler, R.; Kesselman, C.; Koranda, S.; Moe, B. Wide Area Data Replication for Scientific Collaborations. In: *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, Washington, DC, USA: IEEE Computer Society, 2005, p. 1–8.

Choudhury, S.; Bhatnagar, K.; Haque, W. *Public Key Infrastructure Implementation and Design*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

Copeland, L. *A Practitioner's Guide to Software Test Design*. Norwood, MA, USA: Artech House, Inc., 2003.

Cox, M. J.; Engelschall, R. S.; Henson, S.; Laurie, B. OpenSSL Project. Disponível em: http://www.openssl.org/docs/apps/ca.html. Acesso em: 10/01/2008., 2007.

DeFanti, T. A.; Foster, I.; Papka, M. E.; Stevens, R.; Kuhfuss, T. Overview of the I-WAY: Wide-Area Visual Supercomputing. *The International Journal of Supercomputer Applications and High Performance Computing*, v. 10, n. 2/3, p. 123–131, 1996.

Dierks, T.; Karlton, P. L.; Freier, A. O.; Kocher, P. C. The TLS Protocol Version 1.0. http://www.rfc-editor.org/rfc/rfc2246.txt, 1999.

D.W.Chadwick; A. Novikov; A. Otenko GridShib and PERMIS Integration. *Campus-Wide Information Systems*, v. 23, n. 4, p. 297–308, disponível em: http://www.cs.kent.ac.uk/pubs/2006/2530. Acesso em: 01/07/2007., 2006.

Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *J. Comput. Sci. Technol.*, v. 21, n. 4, p. 513–520, 2006.

Foster, I.; Kesselman, C. *The Grid 2: Blueprint for a New Computing infrastructure*. 2 ed. Morgan Kaufmann, 2004.

GridSphere GridSphere Project. Disponível em: http://www.gridsphere.org/. Acesso em: 01/03/2007., 2005.

Housley, R.; Ford, W.; Polk, T.; Solo, D. Internet X.509 Public Key Infrastructure: Certificate and Crl Profile. http://www.rfc-editor.org/rfc/rfc2459.txt, this specification has been superseded by RFC3280, 1999.

Housley, R.; Ford, W.; Polk, T.; Solo, D. Internet X.509 Public Key Infrastructure. http://www.rfc-editor.org/rfc/rfc2246.txt, 2002.

Howes, T.; Kille, S.; Yeong, W.; Robbins, C. The String Representation of Standard Attribute Syntaxes. RFC Editor, 1995.

Humphrey, M.; Wasson, G. Architectural Foundations of WSRF.NET. *Journal of Web Services Research*, disponível em: `http://www.cs.virginia.edu/~humphrey/papers/ArchFoundationsWSRFdotNET.pdf`. Acesso em: 10/06/2007., 2005.

ISO/IEC 10181-3, I. R. . Security Frameworks for open systems: Access Control Framework. Disponível em: `http://www.itu.int/rec/T-REC-X.812-199511-I/en`. Acesso em: 10/08/2007., 1995.

ITU-T The Directory: Public-key and Attribute Certificate Frameworks. 4th ed, recommendation X.509, ISO/IEC 9594-8, 2000.

Jacob, B.; Brown, M.; Fukui, K.; Trivedi, N. *Introduction to Grid Computing*. IBM RedBooks, 262 p., iSBN: 0738494003 e IBM Form Number SG24-6778-00, 2005.

JetSpeed Jetspeed Project. Disponível em: `http://portals.apache.org/jetspeed-2/`. Acesso em: 01/03/2007., 2005.

Klaene, M. Understanding the Java Portlet Specification. Disponível em: `http://www.developer.com/java/web/print.php/10935_3366111_1` . Acesso em: 12/11/2007., 2004.

Kostopoulos, G.; Sklavos, N.; Koufopavlou, O. *Security in Distributed, Grid, and Pervasive Computing, Chapter 10. State-of-the-Art Security in Grid Computing*. CRC-Press, 2007.

Kurose, J. F.; Ross, K. W. *Redes de computadores e Internet: uma abordagem top/down*. 3 ed. Pearson Addison Wesley, tradução Arlete Simille Marques, 2006.

Laszewski, G.; Foster, I.; Gawor, J.; Lane, P. A Java Commodity Grid Kit. Http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf, 2001.

Laszewski, G. V. Commodity Grid (CoG) Kits. Disponível em: `http://wiki.cogkit.org` . Acesso em: 27/12/2007, 2005.

Lento, L. O. B.; Silva Braga, J.; Lung, L. C. A Nova Geração de Modelos de Controle de Acesso em Sistemas Computacionais. In: *Simposio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, 2006, p. 152–201.

Licklider, J. C. R.; Taylor, R. The Computer as a Communication Device. *International Science and Technology*, disponível em: `ftp://gatekeeper.research.compaq.com/pub/DEC/SRC/research-reports/SRC-061.pdf`. Acesso em: 20/06/2007., 1968.

Lingen, F. V.; Steenberg, C.; Thomas, M.; Anjum, A.; Azim, T.; Khan, F.; Newman, H.; Ali, A.; Bunn, J.; Legrand, I. The clarens Web Service Framework for Distributed Scientific Analysis in Grid Projects. In: *International Conference Workshops on Parallel Processing*, IEEE Computer Society, 2005, p. 45–52.

Linn, J. Generic Security Service Application Program Interface, Version 2. http://www.rfc-editor.org/rfc/rfc2078.txt, 1997.

MICROSYSTEMS, S. Introduction to JSR 168 - the Java Portlet Specification. Disponível em: http://developers.sun.com/portalserver/reference/techart/jsr168/pb_whitepaper.pdf. Acesso em: 09/11/2007., 2003.

MITRA, N.; LAFON, Y. SOAP Version 1.2 Part 0: Messaging Framework (Second Edition). Disponível em: http://www.w3.org/TR/2007/REC-soap12-part0-20070427. Acesso em: 11/06/2007., 2007.

MYERS, G. J. *Art of Software Testing*. New York, NY, USA: John Wiley & Sons, Inc., 1979.

NASH, A.; DUANE, W.; JOSEPH, C.; BRINK, D. *PKI: Implementing and Managing E-Security*. Osborne/McGraw-Hill, 2001.

NIST Federal Information Security Management Act - FISMA. Disponível em: http://csrc.nist.gov/groups/SMA/fisma/. Acesso em: 10/06/2007., 2002.

NOVOTNY, J.; RUSSELL, M.; WEHRENS, O. GridSphere's Grid Portlets. In: *Computational Methods in Science and Technology*, 2006, p. 89–97.

NOVOTNY, J.; TUECKE, S.; WELCH, V. An Online Credential Repository for the Grid: MyProxy. in Proc. 10 th IEEE Symp. On High Performance Distributed Computing, 2001.

NPAC, N. P. A. C. FAFNER. Disponível em: www.npac.syr.edu/factoring.html . Acesso em: 11/06/2007, 1995.

NUNES, C. A. A. Collaborative content creation by cross-level students. In: *2th International Conference on Open Collaborative Design for Sustainable Innovation: Creativity, Control & Culture for Sustainable Change*, bangalore, India, 2002, p. 1–2.

OASIS SAML Executive Overview. Committee DRAFT 01, OASIS, 2005.

OASIS Web Services Security: Soap Message Security 1.1. Web Service Security (WSS), disponível em: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf . Acesso em: 01/11/2007, 2006.

OASIS Ws-SecureConversation. Disponível em: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html . Acesso em: 01/11/2007, 2007a.

OASIS WS-Trust 1.3. Disponível em: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf . Acesso em: 01/11/2007, 2007b.

OWASP The Ten Most Critical Web Application Security Vulnerabilities. Disponível em: http://www.owasp.org/documentation/topten.html . Acesso em: 01/04/2008, 2004.

PARK, J. S.; SANDHU, R.; AHN, G. J. Role-based access control on the web. *ACM Trans. Inf. Syst. Secur.*, v. 4, n. 1, p. 37–71, 2001.

PEARLMAN, L.; KESSELMAN, C.; GULLAPALLI, S.; B.F. SPENCER, J.; FUTRELLE, J.; RICKER, K.; FOSTER, I.; HUBBARD, P.; SEVERANCE, C. Distributed Hybrid Earthquake Engineering Experiments: Experiences with a Ground-Shaking Grid Application. In: *HPDC13*, IEEE-P, 2004.

Pearlman, L.; Welch, V.; Foster, I.; Kesselman, C.; Tuecke, S. A Community Authorization Service for Group Collaboration. In: *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, Washington, DC, USA: IEEE Computer Society, 2002, p. 50–59.

Pierce, M.; Gannon, D.; Wilkins-Diehr, N.; Thomas, M.; Edwards, R.; Dahan, M.; Laszewski, G.; Kandaswamy, G.; Marru, S. Open Grid Computing Environment Collaboratory. Disponível em: `http://www.collab-ogce.org/`. Acesso em: 10/06/2007., 2005.

Rizza, J. M. *Computer Network Security*. Springer US, 2005.

Rosatelli, M.; Senger, H.; Silva, F.; Stanzani, S.; Nunes, C. Supporting the Collaborative Construction of Learning Objects Using the Grid. In: *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, Washington, DC, USA: IEEE Computer Society, 2006, p. 257–260.

Roure, D. D.; Baker, M. A.; Jennings, N. R. The Evolution of the Grid. In: *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, disponível em: `http://citeseer.ist.psu.edu/535794.html`. Acesso em: 20/07/2007., 2003, p. 65–100.

Russell, D.; Gangemi, G. *Computer Security Basics*. 1 ed. O'Reilly & Associates, Inc, 464 p., 1991.

Sandhu, R.; Samarati, P. Authentication, Access Control, and Audit. *ACM Computing Surveys*, v. 28, n. 1, 1996.

Sandhu, R. S.; Samarati, P. Access Control: Principles and Practice. *IEEE Communications Magazine*, v. 32, n. 9, p. 40–48, 1994.

Senger, H.; Stanzani, S. L.; Rondini, R. A. Uma Introdução ao Desenvolvimento de Aplicações para Grades Computacionais Baseadas no Padrão OGSA. In: *Mini-Cursos - VII Workshop em Sistemas Computacionais de Alto Desempenho*, VII WorkShop de Sistemas Computacionais de Alto Desempenho, 2006.

Sotomayor, B. The Globus Toolkit 4 Programmers Tutorial. Disponível em: `http://gdp.globus.org/gt4-tutorial`.Acesso em: 24/06/2007, 2005.

Sotomayor, B.; Childers, L. *The Globus Toolkit 4 Programming Java Services*. Morgan Kaufmann Publishers, 2006.

Stanzani, S. L. Um ambiente baseado em Grades Computacionais para suporte a aplicações voltadas a produção de objetos de aprendizagem, dissertação de Mestrado, 2008.

Steenberg, C.; Aslakson, E.; Bunn, J. J.; Newman, H. B.; Thomas, M.; Lingen, F. The Clarens Web Services Architecture. *CoRR*, v. cs.DC/0306002, informal publication, 2003.

Thompson, M. R.; Essiari, A.; Mudumbai, S. Certificate-based authorization policy in a PKI environment. *ACM Trans. Inf. Syst. Secur.*, v. 6, n. 4, p. 566–588, 2003.

Tuecke, S.; Pearlman, L.; Engert, D.; Welch, V.; Thompson, M. Internet X.509 Public Key Infrastruture (PKI) proxy certificate profile. Disponível em: `http://www.rfc-editor.org/rfc/rfc3820.txt` . Acesso em: 15/06/2007., 2004.

uPortal uportal Project. Disponível em: `http://www.uportal.org/`. Acesso em: 01/03/2007., 2005.

Vecchio, D. D.; Hazlewood, V.; Humphrey, M. Evaluating Grid Portal Security. In: *SC'06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, New York, NY, USA: ACM Press, 2006, p. 114.

Welch, V. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective, disponível em: `http://www.globus.org/toolkit/docs/4.0/security/`.Acesso em: 02/06/2007, 2005.

Welch, V.; Foster, I.; Kesselman, C.; Mulmo; Pearlman, L.; Tuecke, S.; Gawor, J.; Meder, S.; Siebenlist, F. X.509 Proxy Certificates for Dynamic Delegation. *3rd Annual PKI R&D Workshop*, 2004.

Zhang, C.; Kelley, I.; Allen, G. Grid Portal Solutions: a Comparison of Gridportlets and OGCE. *Concurr. Comput. : Pract. Exper.*, v. 19, n. 12, p. 1739–1748, 2007.

Zhu, L.; Kent, R. D.; Aggarwal, A.; Viranthi, P.; Rahman, Q.; Elamsy, T.; Ejelike, O. Construction of a Webportal and User Management Framework for Grid. In: *HPCS '07: Proceedings of the 21st International Symposium on High Performance Computing Systems and Applications*, Washington, DC, USA: IEEE Computer Society, 2007, p. 14.

Apêndice A

Administração e gerenciamento do IAGP

Conforme descrito no capítulo 4, o uso do IAGP adiciona várias funcionalidades não existem na solução de portal de grade original (GridSphere/GT4). A Figura A.1 mostra a interface apresentada para o usuário durante o processo de autenticação no portal GridSphere. Qualquer usuário registrado no portal com o IAGP instalado necessita definir um identificador de usuário, uma senha e um papel para ter sua identidade verificada pelo portal (componentes IAGP-Login e IAGP-Auth).

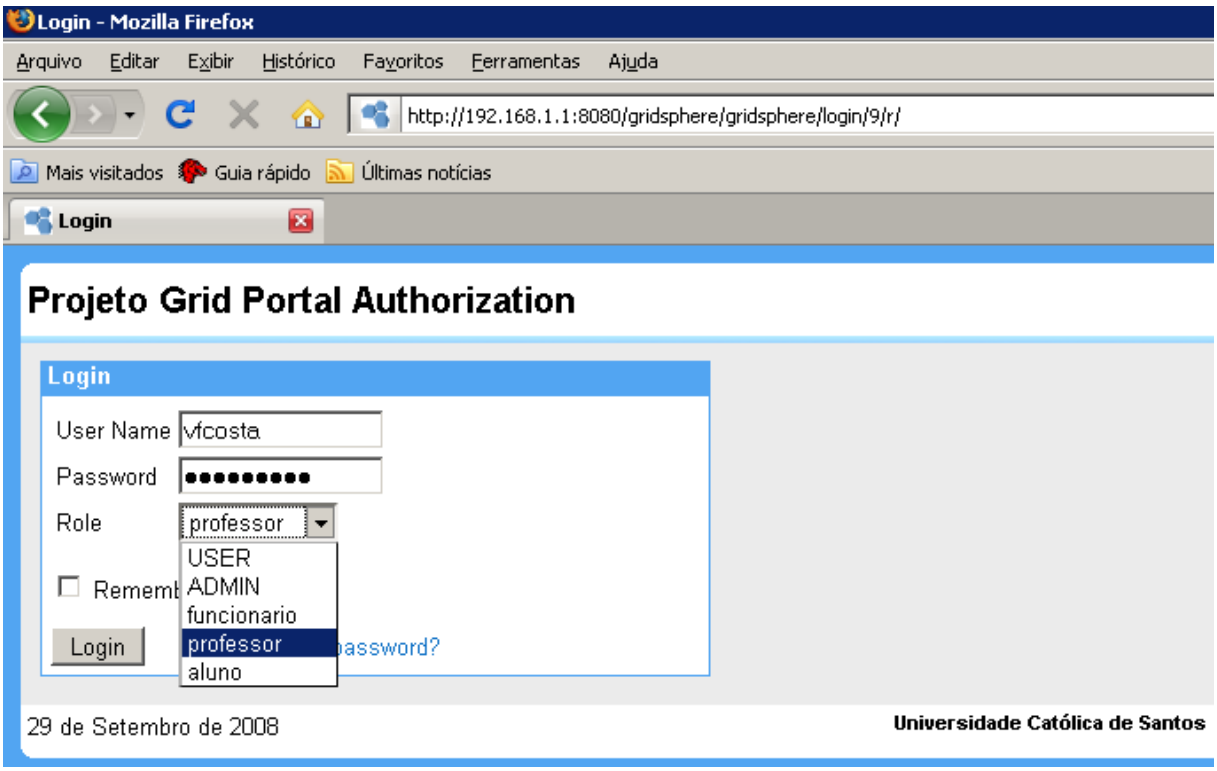

Figura A.1: Nova interface para autenticação dos usuários oferecida pelo IAGP

Nesse cenário, os papéis foram adicionados com o uso de recurso próprio do portal GridSphere conforme ilustra a Figura A.2.

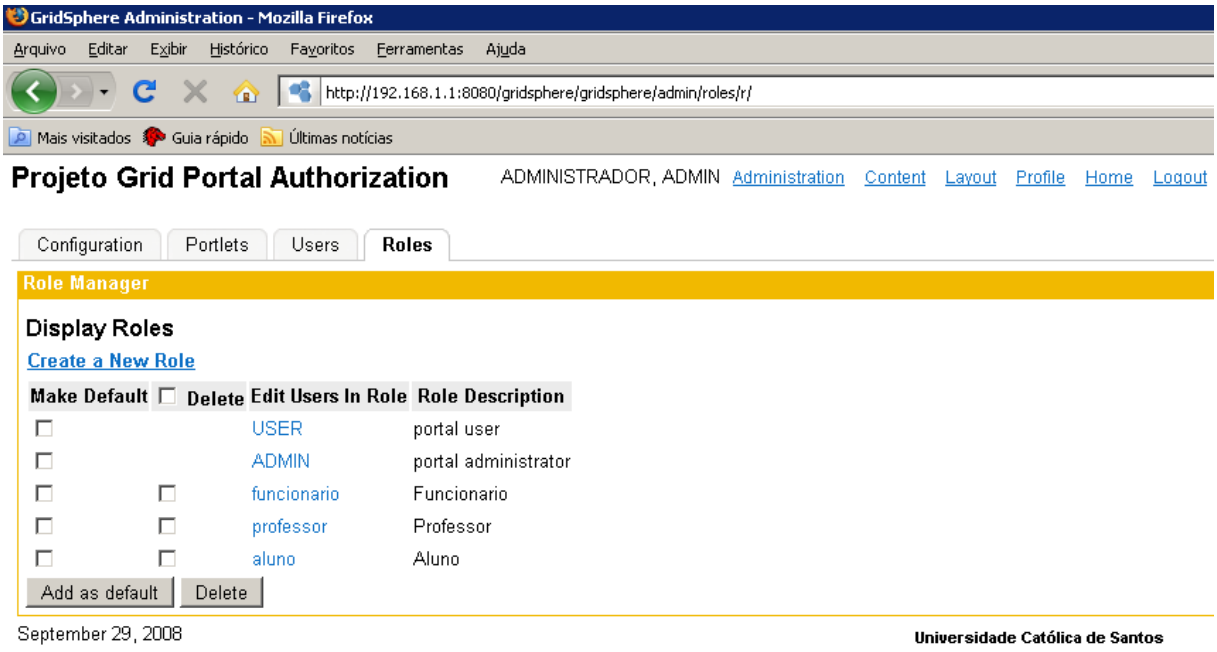

Figura A.2: Lista de papéis registrados no portal.

A Figura A.3 ilustra os usuários registrados no portal pelo administrador.

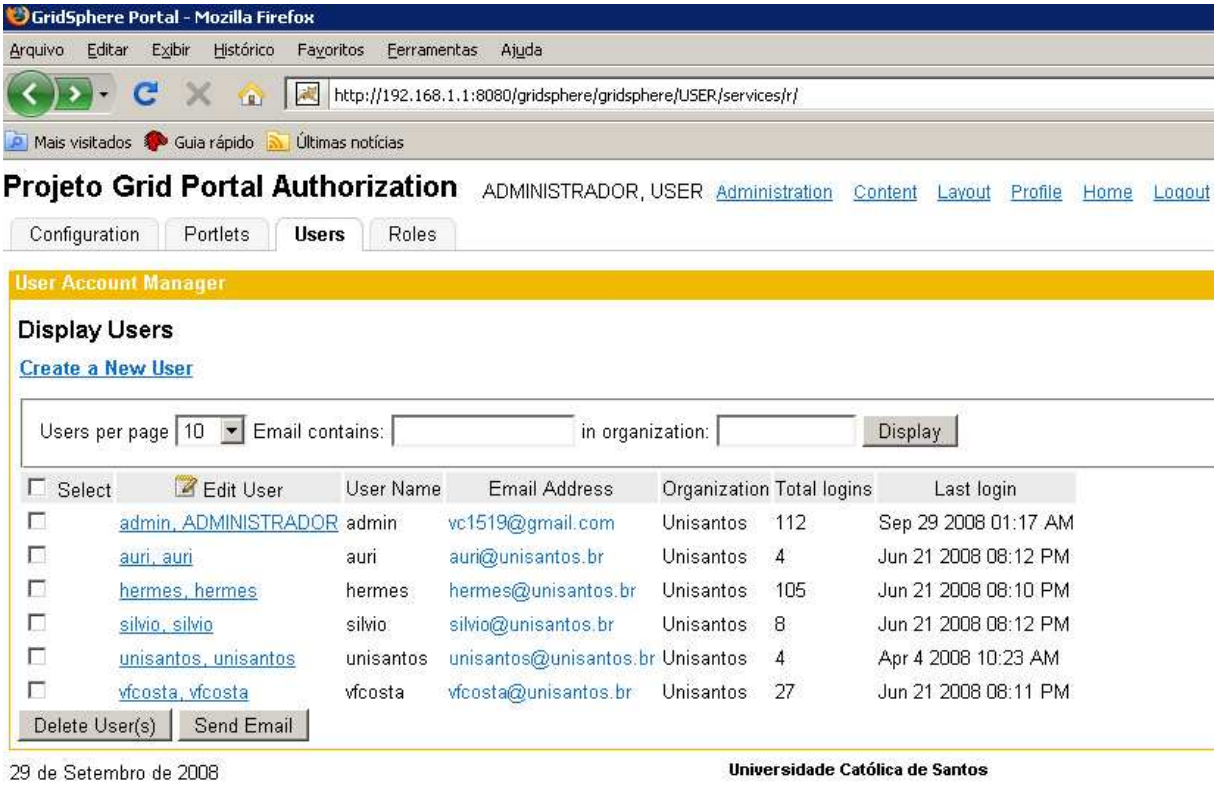

Figura A.3: Lista de usuários cadastrados no portal GridSphere

A associação do usuário a papéis é definida pelo administrador no GridSphere conforme ilustra a Figura A.4.

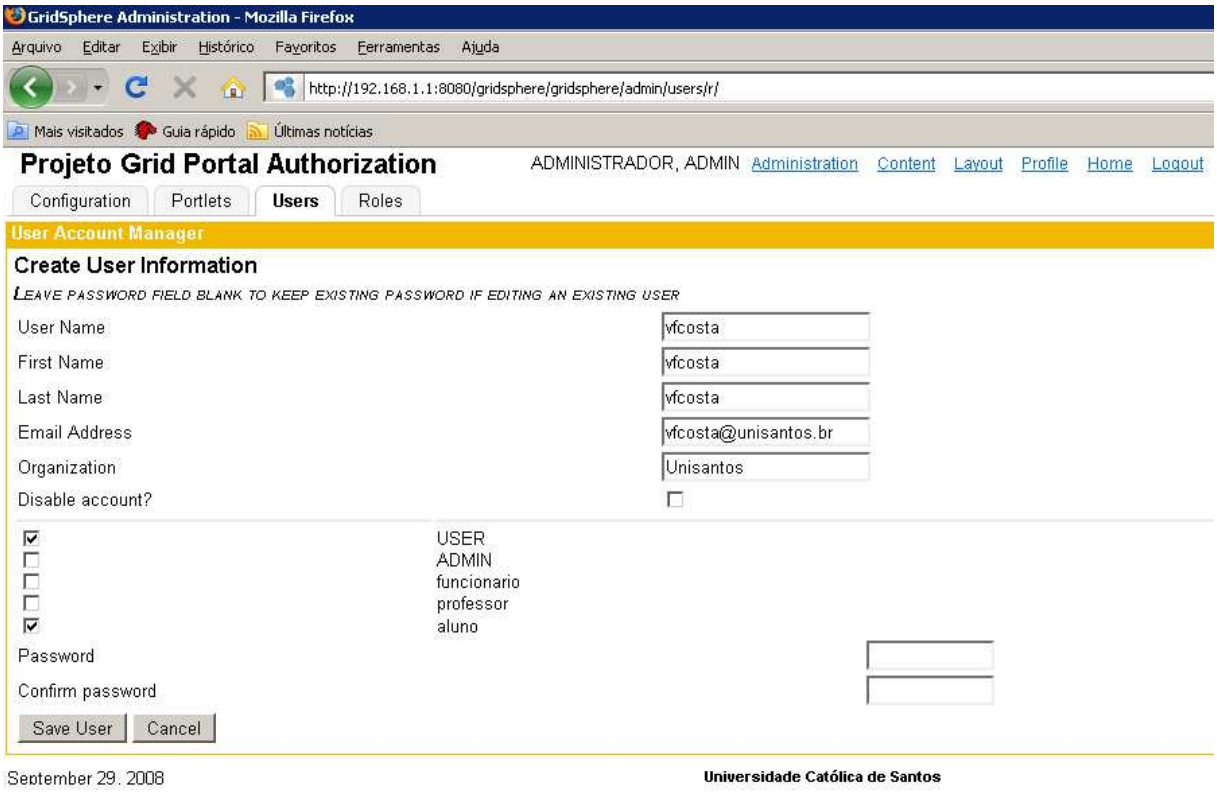

Figura A.4: Definição de papéis para o usuário no GridSphere

O gerenciamento dos serviços a serem protegidos pelo IAGP é ilustrado pela Figura A.5.

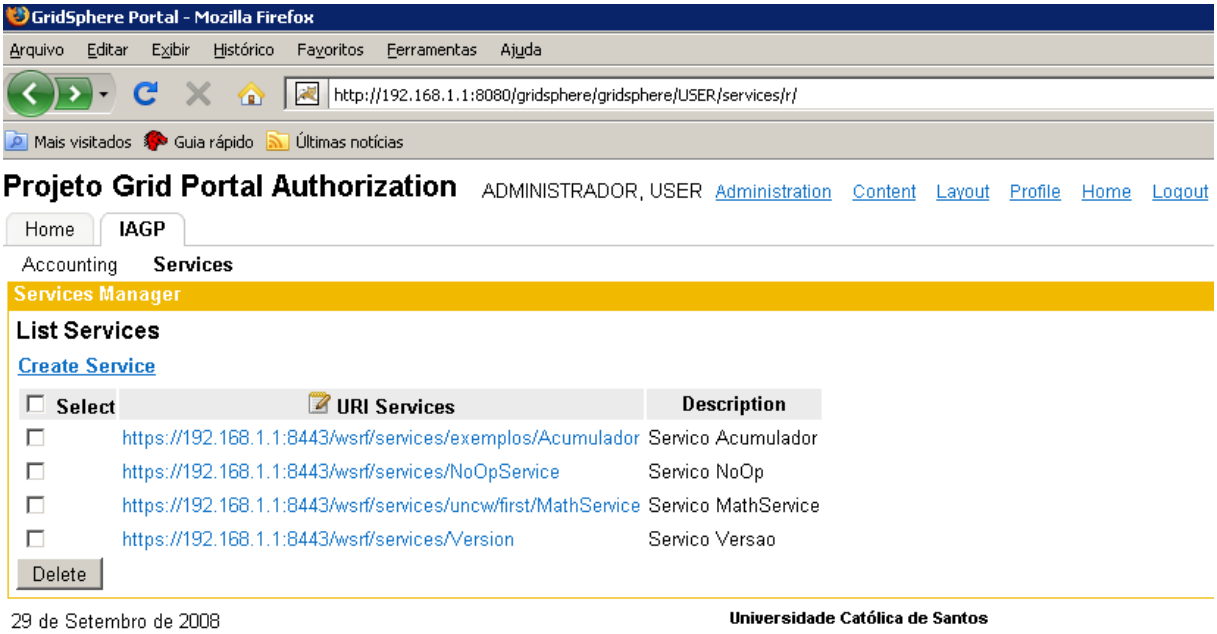

Figura A.5: Serviços a serem protegidos pelo IAGP

A associação do papel requerido para acessar o serviço pode ser feita pelo administrador ao clicar na lista de serviços protegidos pelo IAGP conforme a Figura A.6..

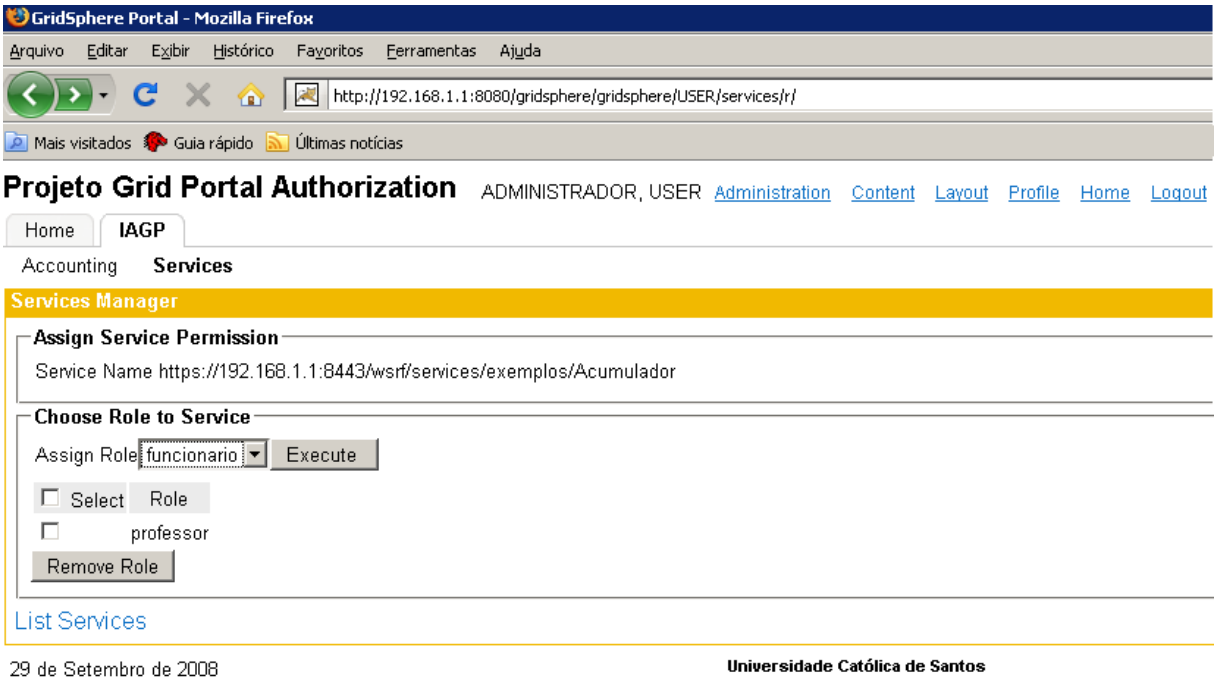

Figura A.6: Associação do papel ao serviço pelo IAGP

A Figura A.7 mostra os registros de eventos de atividades dos usuários integrados pelo IAGP.

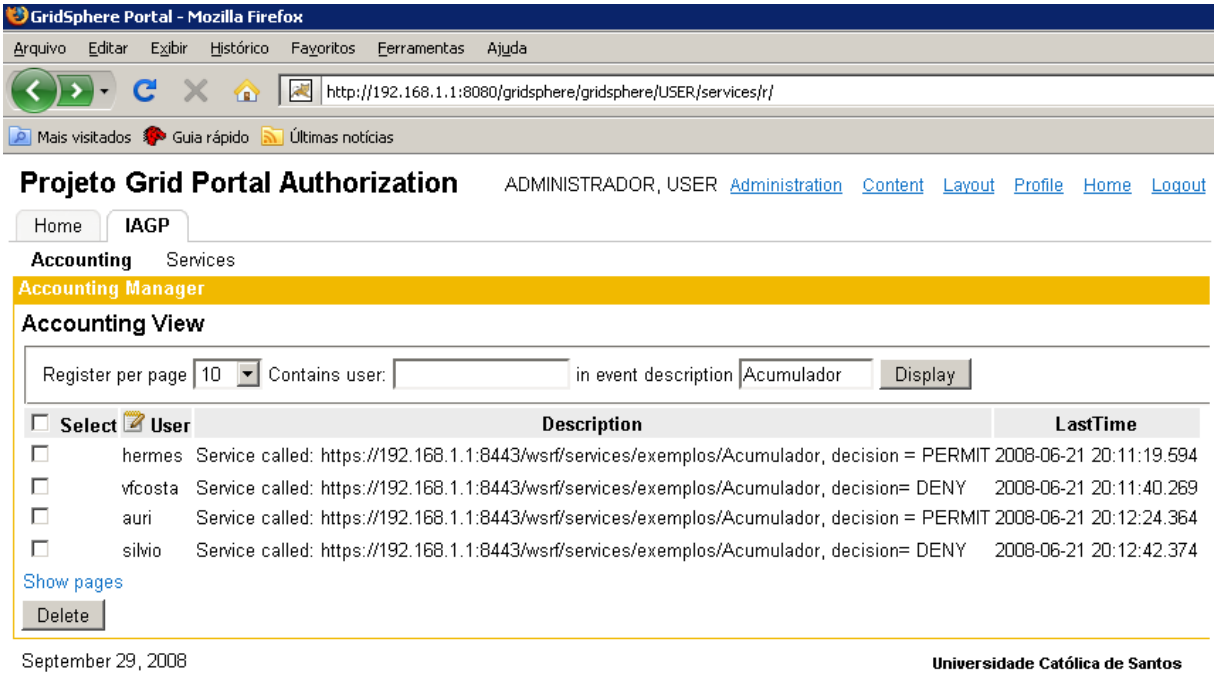

Figura A.7: Relatório integrado de atividades dos usuários do IAGP

Apêndice B

Padrão JSR-168

Nos últimos anos, a tecnologia de portais foi adotada por um grande número de entidades para hospedar suas aplicações. No passado, cada fabricante de soluções para portal definia sua própria API para construção de portais e seus componentes. A falta de padronização impossibilitava a portabilidade das aplicações desenvolvidas entre portais de diferentes fabricantes e limitava o potencial do mercado de portais. A necessidade de uma padronização e a interoperabilidade entre os portais e o *portlets*, levou a aprovação do padrão JSR-168 para desenvolvimento de portais e seus componentes pelo *Java Community Process* (JCP). JCP é a entidade responsável de definição dos padrões utilizados na especificação Java.

A versão 1.0 da especificação JSR-168 foi homologada em outubro de 2003. A especificação JSR-168 define um conjunto de API para o desenvolvimento de *portlets* e um contrato entre o contêiner de *portlets* e um *portlet*. JSR-168 é baseada na especificação de Java *Servlet* API. O comportamento dos *portlets* é semelhante aos *servlets* em várias maneiras. Por exemplo, ambos são componentes *web* baseados em Java, gerenciados por um contêiner especializado e utilizado para gerar conteúdo dinâmico e interativo para clientes *web*. Os *portlets* têm algumas características adicionais em relação aos *servlets*. Como por exemplo, os *portlets* não geram páginas HTML completa, mas pequenos trechos do código de linguagens de marcação, tais como: HTML, WML e XHTML (Akram et al., 2005). *Portlets* são compatíveis com o padrão J2EE. Semelhantes a *servlets*, os *portlets* têm um ciclo de vida gerenciado por um contêiner através de alguns métodos, tais como:

- init(): Esse método é chamado apenas uma vez, imediatamente após uma nova instância do *portlet* ter sido criada. Configurações especiais são passadas a esse método através de um arquivo XML chamado de descritor de *portlets* (portlet.xml).
- destroy(): Esse é o método chamado para encerrar o ciclo de vida do *portlet*, removendo-o da memória.
- processAction(): Esse método o responde com acesso a camada de negócios a partir das ações do usuário como por exemplo, ao submeter dados em um formulário.
- render(): Esse método é chamado a todo o momento para que o *portlet* seja redesenhado na tela.

A Figura B.1ilustra o ciclo de vida de um *portlet*.

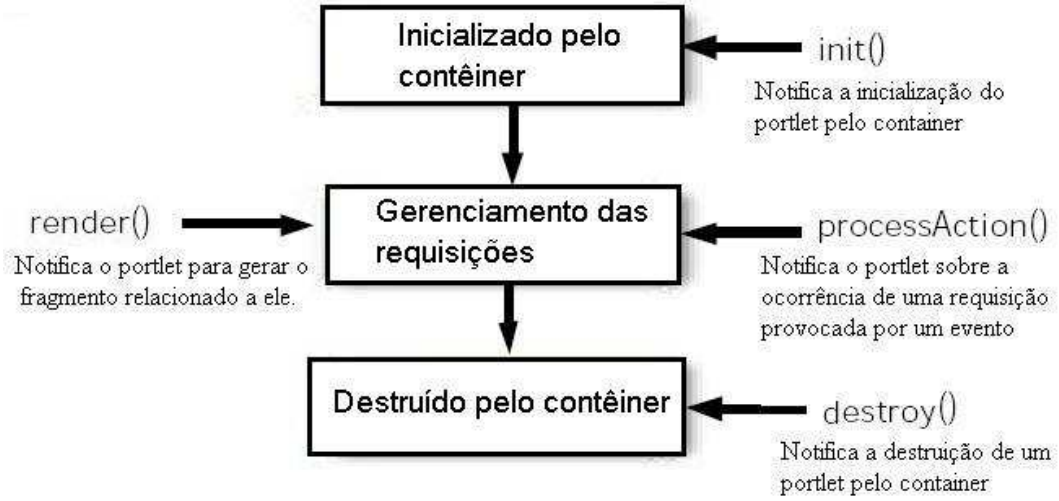

Figura B.1: Ciclo de vida de um *portlet*

O contêiner também gerencia outros controladores que representam o estado de um *portlet*: o modo (*mode*) e a janela (*window state*). O modo determina que ações possam ser realizadas com um determinado *portlet*, tais como: *View*, *Edit* e *Help*. Diferentemente de um *servlet*, cada *portlet* tem uma janela que determina o quanto de conteúdo de um *portlet* pode estar disponível para o usuário. De maneira semelhante a uma aplicação gráfica, a janela de um *portlet* pode assumir três estados: normal, minimizado e maximizado.

A especificação JSR-168 define poucos métodos que envolve armazenamento de informações do usuário, de forma permanentemente ou por sessão. Dois importantes conceitos

são *PortletPreferences* e *PortletSession*. São eles que ajudam o desenvolvedor a compartilhar valores entre os componentes da aplicação do usuário ou alterar as preferências de cor ou *layout* da tela. *PortletPreferences* é um objeto que pode ser usado para armazenar dados persistentes para o *portlet* do usuário. *PortletPreferences* armazena uma série de variáveis no formato nome e valor que podem ser alteradas e estarão disponíveis mesmo após o reinício do serviço ou término da sessão do usuário. Por exemplo, com esse recurso o portal *grid* pode alterar a configuração do ambiente do usuário.

Um *portlet* além de herdar um descritor J2EE para uma aplicação *web*, definido pelo arquivo web.xml, um novo descritor foi adicionado a especificação JSR-168 chamado de portlet.xml. O arquivo portlet.xml define todos os *portlets* e a configuração relacionada entre eles. Um *Schema* XML é incluído na especificação que define os elementos padrão para configuração dos *portlets* registrados no arquivo portlet.xml (Microsystems, 2003). Conforme comentado no inicio desse capítulo, esse descritor é utilizado pelo método init() para configurar o *portlet*.

Um exemplo do arquivo descritor do portal, portlet.xml é mostrado a seguir:

```
<portlet-app>
 <portlet>
   <portlet-name>HelloWorldPortlet</portlet-name>
   <portlet-class>br.unisantos.portlet.HelloWorldPortlet</portlet-class>
   <init-param>  --Parâmetros para o método init()
   <name>view-to-present
 <value>/portlet/HelloWorldPortlet/startup_view.jsp</value>10:55 14/11/2007
   </init-param>
   <expiration-cache>300</expiration-cache> --Expiração do cache (5 minutos)
    <supports>
     <mime-type>text/html</mime-type>     --Suporte HTML
       <portlet-mode>VIEW</portlet-mode>  --Suporte mode View
       <portlet-mode>EDIT</portlet-mode>  --Suporte mode Edit
    </supports>
    <resource-bundle>
```

```
        br.unisantos.portlet.MeuResourceBundle
    </resource-bundle>
    <portlet-preferences>
     <preference>
       <name>Country1</name> -- Preferências nome/valor
       <value>USA</value>
     </preference>
     <preference>
        <name>Country2</name>
        <value>Japan</value>
     </preference> -- Verificador das preferências configuradas
     <preferences-validator>
          br.unisantos.checkit.Validador
     </preferences-validator>
    </portlet-preferences>
   </portlet>
</portlet-app>
```

A seqüência de eventos abaixo é típica quando os usuários acessam o portal, conforme a Figura B.2:

- Um cliente (navegador *web*) após ter sido autenticado faz uma requisição HTTP para o portal.
- A requisição é recebida pelo portal.
- O portal determina se a requisição contém uma ação direcionada para qualquer um dos *portlets* associado com a página do portal.
- Se há uma ação direcionada para um *portlet*, o portal requisita para o contêiner de *portlet* invocar o *portlet* para processar a ação.
- Um portal invoca *portlets*, através do contêiner de *portlets* para obter o conteúdo que pode ser incluído no resultado da página do portal.

- O portal agrega a saída dos *portlets* dentro da página do portal e envia a página do portal de volta para o cliente.

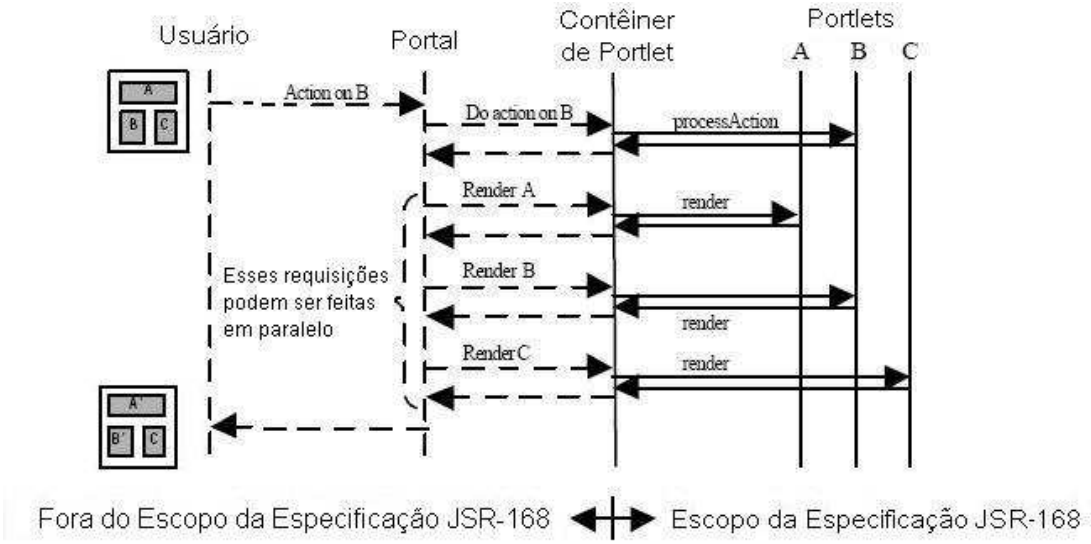

Figura B.2: Interação do usuário com um portal, adaptado de Microsystems (2003)

Muitas ferramentas de portais suportam o padrão JSR-168, tais como:

- GridSphere: Projeto *open-source* europeu, que foi um dos primeiros portais desenvolvidos em conformidade com o padrão JSR-168 (GridSphere, 2005).

- Jetspeed: é uma implementação *open-source* de um arcabouço de portais corporativos que utiliza o Pluto (implementação de referência da especificação JSR-168) como seu contêiner de *portlets* (JetSpeed, 2005).

- uPortal: Muito difundido na comunidade acadêmica e utilizado por várias instituições (uPortal, 2005).

Segurança: A especificação JSR-168 inclui diversos itens que ajudam o desenvolver *portlet*(s) seguros. Dentro do descritor do *portlet*(portlet.xml) um alerta (*flag*) pode ser configurado para executar somente sobre HTTPS. Isso é apropriado para *portlets* que contém informação confidencial que deva ser cifrada quando enviada pela rede. A API dessa especificação inclui facilidades para pesquisar informação de usuário e papéis. Isso permite aos desenvolvedores, através de programação, controlar a segurança pela introdução da lógica de negócio baseada na informação do usuário e papéis. De forma similar a tecnologia J2EE, a especificação de *portlets* não se preocupa como os usuários ou papéis

são implementados, deixando para os fabricantes de portais definirem sua própria solução. Esse é um recurso explorado pelos portais para definir entre os itens, a permissão requerida para executar o *portlet*. Por exemplo, uma aplicação com *portlets* contém recursos que podem ser acessados por vários usuários. Freqüentemente esses recursos atravessam redes desprotegidas e abertas como a Internet. Nesse ambiente, um considerável número de aplicações deverão atender a um conjunto de requisitos (autenticação, autorização e rastreabilidade) relacionados de área de segurança da informação. O contêiner de *portlets* é o responsável por gerenciar usuários, grupos, papéis e permissões. Essa informação combinada com métodos da API do *Servlet*, descrito abaixo, permitem determinar a identidade do usuário e o papel associado a ele. Dessa maneira é possível criar um mecanismo de controle baseado em papéis para qualquer portal que esteja em conformidade com o padrão JSR-168. O portal GridSphere é um exemplo da aplicação desse recurso. A especificação JSR-168 compartilha a mesma definição de papéis da especificação de *servlets* (*Servlet* 2.3, sessão SRV.12.4). Essa API permite criar as decisões sobre a lógica de negócio de uma aplicação através da informação obtida por esses métodos (Abdelnur e Hepper, 2003), citados a seguir:

- getRemoteUser: Se o usuário está autenticado, retorna o identificador do usuário que faz uma requisição ao serviço *web*, caso contrário, retorna o valor nulo.

- isUserInRole: Esse método determina se um usuário está associado com um papel em particular e retorna um valor verdadeiro ou falso. Se o usuário não está autenticado, retorna o valor falso.

- getUserPrincipal: Se o usuário está autenticado, retorna determina o identificador da entidade corrente. Caso o usuário não esteja autenticado, retorna o valor nulo.

Resumindo, o padrão J2EE herdado pela especificação JSR-168 oferece suporte a um sistema de autorização baseado em papéis. Essa especificação, também oferece um descritor baseado em um arquivo XML que define a permissão requerida para acessar um recurso em particular.

Apêndice C

Descritores de Segurança

Os descritores de segurança contém várias propriedades como localização do arquivo *gridmap*, requerimentos para mecanismos de autenticação e autorização, credenciais, etc. Há quatro tipos descritores de segurança para configurar propriedades do contêiner, serviço, recurso e do cliente que são detalhados a seguir:

- contêiner: determina os requerimentos de segurança que deve ser aplicado em nível do contêiner.
- serviço: determina os requerimentos de segurança que deve ser aplicado em nível de serviço.
- recurso: determina os requerimentos de segurança que deve ser aplicado em nível do recurso.
- cliente: determina as propriedades que devem ser usadas para uma particular invocação.

Cada um desses descritores está representado como um objeto e pode ser alterado através de programação. O descritor do serviço e o descritor do contêiner podem ser configurados como arquivos XML dentro de um arquivo WSDD (*deployment descriptor*). O descritor de segurança de recurso pode somente ser criado dinamicamente, ou seja, não pode ser configurado nem utilizando programação ou nem de arquivo de configuração XML. Já um descritor de segurança de cliente pode ser configurado através de um arquivo

XML e configurando as propriedades de um *Stub*. Se o descritor do cliente sofrer alguma alteração em tempo de execução, o programa deve ser reiniciado (Alliance, 2006).

Se o descritor de segurança está configurado para ser lido de um arquivo, ele é lido como segue:

- Como um arquivo, se o endereço absoluto é especificado.
- Como um recurso que pode ser incluído como parte de um pacote java.
- Como um arquivo, considerando um caminho especificado relativo ao diretório raiz da instalação, tipicamente configurado pela variável GLOBUS_LOCATION.

Os arquivos do descritor de segurança necessitam respeitar as regras do esquema do descritor de segurança de serviço ou do contêiner de maneira adequada. O arquivo do descritor de segurança de recurso utiliza o mesmo esquema do descritor de serviço. Em todos os casos, o descritor de segurança é contido dentro do elemento <securityConfig xmlns="http://www.globus.org">.

O arcabouço irá primeiro procurar a configuração *gridmap* configurada pelo recurso, depois o serviço e, por último, o contêiner. Para que os serviços configurados obtenham a autorização por meio do arquivo *gridmap*, existe uma API que permite que ele seja atualizado dinamicamente (SecurityManager).

C.1 Descritores de Contêiner

O descritor de segurança do contêiner é configurado através do arquivo $GLOBUS_-LOCATION/etc/globus_wsrf_core/server-config.wsdd que contém o seguinte trecho de código da Figura C.1:

```
<globalConfiguration>
   ...
   <parameter name="containerSecDesc"
              value="etc/container-security-config.xml">
   ...
<globalConfiguration>
```

Figura C.1: Descritor de segurança de contêiner (Alliance, 2006)

O arquivo $GLOBUS_LOCATION/etc/container-security-config.xml contém os parâmetros necessários para localizar o certificado e a chave privada que vai iniciar o contêiner do GT4 e a localização do arquivo *grid-mapfile*.

Exemplo A

```
<securityConfig xmlns="http://www.globus.org">
   ...
   <credential>
      <key-file value="keyFile"/>
      <cert-file value="certFile"/>
   </credential>
   ...
</securityConfig>
```

Exemplo B

```
<securityConfig xmlns="http://www.globus.org">
   ...
   <proxy-file value="proxyFile"/>
   ...
</securityConfig>
```

Figura C.2: Configuração do descriptor do lado servidor (Alliance, 2006)

C.2 Descritores de Serviço

O descritor de segurança do serviço pode ser feita da mesma forma do exemplo da Figura C.3:

```
<service name="MyDummyService" provider="Handler" style="document">
   ...
   <parameter name="securityDescriptor"
value="org/globus/wsrf/impl/security/descriptor/security-config.xml"/>
   ...
</service>
```

Figura C.3: Descritor de segurança de serviço (Alliance, 2006)

C.3 Descritores de Cliente

O descritor de um cliente pode ser configurado de acordo com o exemplo da Figura C.4: O descritor de segurança necessita atender ao esquema do descritor de segurança contido do cliente.

```
// Client security descriptor file
String CLIENT_DESC = "org/globus/wsrf/samples/counter/client/client-security-config.xml";
//Set descriptor on Stub
((Stub)port)._setProperty(Constants.CLIENT_DESCRIPTOR_FILE, CLIENT_DESC);
```

Figura C.4: Descritor de segurança do cliente (Alliance, 2006)

C.4 Configurando o servidor

O contêiner e cada serviço do GT4 pode ser configurado com um separado conjunto de credenciais. As credenciais podem ser configuradas ou apontando para a localização do certificado e da chave privada ou para o arquivo *proxy*. Se alguns desses arquivos em tempo de execução sofrer modificação, as credenciais serão automaticamente recarregadas. As credenciais podem ser configuradas adicionando o bloco de código a seguir para um contêiner ou serviço.

Para configurar o descritor de segurança que permita a inicialização de um contêiner ou serviço através de um certificado X.509 é necessário a configuração abaixo:

```
<securityConfig xmlns=''http://www.globus.org''>
   ...
   <credential>
      <key-file value=''keyFile''/>
      <cert-file value=''certFile''/>
   </credential>
   ...
</securityConfig>
```

E para o caso de se preferir utilizar um certificado *proxy*, na configuração do descritor será necessária a seguinte alteração:

```
<securityConfig xmlns=''http://www.globus.org''>
   ...
   <proxy-file value=''proxyFile''/>
   ...
</securityConfig>
```

O container para cada serviço pode ser configurado com um separado gridmap definido no descritor como:

```
<securityConfig xmlns=''http://www.globus.org''>
   ...
   <gridmap value=''gridMapFile''/>
   ...
</securityConfig>
```