LEONARDO MATTES

**GRIDMULTIPOLICY: GERENCIAMENTO E EFETIVAÇÃO DE
MÚLTIPLAS POLÍTICAS DE CONTROLE DE ACESSO EM
AMBIENTES DE GRADES COMPUTACIONAIS**

**São Paulo
2007**

LEONARDO MATTES

**GRIDMULTIPOLICY: GERENCIAMENTO E EFETIVAÇÃO DE
MÚLTIPLAS POLÍTICAS DE CONTROLE DE ACESSO EM
AMBIENTES DE GRADES COMPUTACIONAIS.**

**São Paulo
2007**

LEONARDO MATTES

**GRIDMULTIPOLICY: GERENCIAMENTO E EFETIVAÇÃO DE
MÚLTIPLAS POLÍTICAS DE CONTROLE DE ACESSO EM
AMBIENTES DE GRADES COMPUTACIONAIS.**

Tese apresentada à Escola Politécnica
da Universidade de São Paulo para
obtenção do Título de Doutor em
Engenharia.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. João Antônio Zuffo

São Paulo
2007

FICHA CATALOGRÁFICA

Mattes, Leonardo

**GridMultiPolicy:Gerenciamento e efetivação de múltiplas políticas de controle de acesso em ambientes de grades computacionais / Leonardo Mattes -- São Paulo, 2007.
123 p.**

Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

**1. Redes de computadores (segurança; gerenciamento)
2. Grades Computacionais 3. Políticas de segurança
I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t**

AGRADECIMENTOS

Primeiramente a Andreza, por servir de fonte de inspiração e encher minha vida com cores e brilhos.

Aos meus pais Raquel e Delmar e minha irmã Anita, pelos anos de compreensão, incentivo, ajuda e por valorizarem meus estudos.

Ao meu orientador Prof. Dr. João Antônio Zuffo, pela orientação oferecida de forma atenciosa e por toda a ajuda a mim prestada.

Aos meus amigos Quinho, Luiz e Mauricio, pelos conselhos e pela amizade, companheiros presentes nos momentos felizes e difíceis, que desempenharam papel indireto, porém fundamental, nesta empreitada.

Aos meus amigos Leonardo e Matteo, pela amizade e por me receberem como membro da equipe, ajudando-me por meio de troca de idéias, conselhos e sugestões que enriqueceram muito o meu trabalho.

Aos que me ajudaram na realização desse trabalho, entre os quais destaco Maria, Leonardo e Delmar.

À agência CAPES pelo financiamento do presente trabalho.

RESUMO

O termo grade computacional faz referência a uma classe de sistemas distribuídos que permitem a associação e a integração de múltiplos domínios em organizações virtuais. Um serviço de controle de acesso coerente a sistemas com estas características deve ser flexível para integrar múltiplas políticas, permitindo que administradores, sítios e usuários determinem as regras e os mecanismos para proteger seus recursos. Esta tese apresenta o GridMultiPolicy, um sistema flexível para o gerenciamento e a integração de múltiplas políticas e mecanismos para a efetivação de controle de acesso em ambientes de grade computacional. Adicionalmente, foram desenvolvidas políticas para demonstrar a capacidade do sistema proposto em oferecer respostas às necessidades presentes em cenários de uso de uma grade computacional. O impacto da utilização do GridMultiPolicy e das políticas desenvolvidas foi avaliado por meio de testes de desempenho.

Palavras-chave: Grade computacional. Grades. Sistemas distribuídos. Políticas de segurança em organizações virtuais. GridMultiPolicy.

ABSTRACT

The term computational grid refers to a class of distributed systems that allows for the association and integration of multiple independent domains in virtual organizations. A coherent access control services in such a system should be flexible to integrate multiple policies so as to permit administrators, sites, and users to determine roles and mechanisms to protect their resources. This thesis introduces the GridMultiPolicy, a flexible system that manages and integrates multiple policies and mechanisms to enforce access control in grid environments. Additionally, policies have been developed to show how the proposed system is able to offer answers to security needs present in grid use scenarios. The impact of the GridMultiPolicy and the developed policies was evaluated by performance tests.

Key-words: Computational grid. Grid. Distributed system. Virtual organization. Policy and security for virtual organization. GridMultiPolicy.

LISTA DE ILUSTRAÇÕES

Figura 1 - Modelo de autorização <i>XACML</i>	34
Figura 2 – Certificado digital.....	37
Figura 3 - Modelo de certificado <i>SPKI</i> (WANG, 1998).	39
Figura 4 - Processo de delegação de autenticação utilizando <i>MyProxy</i>	40
Figura 5 - Modelo de camadas da plataforma <i>Globus Toolkit</i> (FOSTER, 2005).	44
Figura 6 - Estrutura dinâmica de relações de confiança das Organizações Virtuais envolvendo múltiplos domínios (WELCH et al., 2003).	46
Figura 7 - Modelo de segurança <i>OGSA</i> (NAGARATNAM et al., 2002).	48
Figura 8 - Exemplo de credencial de acesso contendo a chave pública e atributos de segurança do usuário (PEARLMAN et al., 2003).	50
Figura 9 – Arquitetura <i>InteGrade</i>	53
Figura 10 – Uso da <i>CAS</i> no processo de autorização (PEARLMAN et al., 2003).	59
Figura 11 – Seqüência do processo de autorização <i>Akenti</i> (Thompson et al. 2003).	60
Figura 12 – Exemplo de estrutura com múltiplas políticas.	61
Figura 13 – Modelo de controle de acesso semelhante aos propostos por MATTES e ZUFFO (2006) e PARK e HUMPHREY (2006).	63
Figura 14 – Modelo de controle de acesso genérico da plataforma <i>AAA</i>	68
Figura 15 – Problemática de diferentes níveis de granularidade.....	69
Figura 16 – Modelo de segurança <i>GridMultiPolicy</i>	69
Figura 17 – Modelo de gerenciamento de políticas em múltiplos formatos.....	70
Figura 18 – Privilégio efetivo, a interseção entre os privilégios da conta mapeada e a da política estabelecida.....	72
Figura 19 – Modelo de certificado de políticas.	73
Figura 20 – Posicionamento da camada de efetivação.	75
Figura 21 – Estrutura hierárquica do <i>PDP</i>	75
Figura 22 – Arquitetura geral do <i>GridMultiPolicy</i>	78
Figura 23 – Estabelecimento de ambiente de execução configurado.	79
Figura 24 – Ponto de efetivação de política da plataforma <i>Java</i>	80
Figura 25 – Código do método "invoke()" da classe <i>org.apache.axis.client.Call</i> a) original sem modificações b) após modificações que incluem o <i>PEP</i> para	

granularidade de serviços web.....	82
Figura 26 – Instrução para criação do PEP na biblioteca Axis.....	83
Figura 27 – Efetivação das Políticas de Segurança.....	84
Figura 28 – Interface de comunicação PDP “br.usp.lsi. JMPE.PDPInterface”.....	85
Figura 29 – Algoritmo de resolução de conflito Preferência pela Negação (XACML 2007).....	86
Figura 30 – Algoritmo de resolução de conflito Preferência pela Autorização (XACML 2007).....	87
Figura 31 – Algoritmo de resolução de conflito Primeiro Aplicável (XACML 2007).....	87
Figura 32 – Efetivação de uma regra ‘ <i>Restrict</i> ’.....	89
Figura 33 - Modelo da linguagem de política proposto.....	91
Figura 34 - Elemento “Decision” do XACML modificado.	92
Figura 35 - Escopo da política “webservice”.....	93
Figura 36 - Regra de escopo que define o uso de políticas em sítios do domínio “lsi.usp.br”	94
Figura 37 - Regra de escopo que define o uso de políticas para controlar tarefas e serviços que requisitarem o uso de banco de dados.	94
Figura 38 – Serviço de distribuição de políticas de múltiplos formato.	95
Figura 39 - Interface “PolicyInterface” para a comunicação entre o módulo principal e os secundários, na geração da credencial de acesso.....	95
Figura 40 – Processo de submissão de tarefas.	96
Figura 41 – Processo de submissão de serviços.	97
Figura 42 – Processo de autorização <i>no Java WS Core</i>	98
Figura 43 - Rede corporativa com sistema IDS distribuído.	100
Figura 44 - Ambiente de grade computacional para laboratórios virtuais (FUSIONGRID, 2007).	102
Figura 45 – Modelo de controle de acesso para recursos locais específicos.....	103
Figura 46 – Controle de acesso em serviços da rede.	104
Figura 47 - Exemplo de declaração de privilégios de perfil.	105
Figura 48 – Instrução para criação do PEP em serviços web na biblioteca Axis ..	106
Figura 49 – IDS_Control integrado em uma infra-estrutura de IDS/IPS.....	108
Figura 50 – Instrução para a criação do PEP para a ação “DBConnect”.	109
Figura 51 – Instrução para a criação do PEP para a ação “ExecuteSQL”.	110
Figura 52 – Exemplo de política DBControl.....	110

Figura 53 – Exemplo de equipamentos disponibilizados pela Universidade de Illinois para a realização de experimentos remotos via grade computacional (FUSIONGRID, 2007).	111
Figura 54 – Rede corporativa da lab.Material.....	112
Figura 55 – Instrumento para execução de testes sobre o comportamento de estruturas.	114
Figura 56 – Descrição do arquivo de tarefa “test_PDP.rsl”	116
Figura 57 – Resultado da execução da tarefa “test_PDE.rsl”.....	118
Figura 58 – Rede estabelecida para a realização de testes.....	120
Figura 59 – Tempo médio de instanciação remota de tarefas utilizando diferentes tamanhos de políticas.	123
Figura 60 – Tempo para escrita em arquivos locais com e sem a utilização do GridMultiPolicy.	124
Figura 61 – Sobrecarga proporcional na utilização de conexões <i>TCP/IP</i>	125
Figura 62 – Sobrecarga proporcional no acesso a banco de dados.	126
Figura 63 – Tempo para iniciação do contêiner no <i>Java WS Core</i>	127
Figura 64 – Tempo para criação de contextos.	128

LISTA DE QUADROS

Quadro 1 - Exemplo de utilização da linguagem lógica para representar modelos de derivação de privilégios.....	31
Quadro 2 - Exemplo de utilização da linguagem lógica para representar resoluções de conflitos.	32
Quadro 3 – Quadro Comparativo entre as infra-estruturas ICP e PMI (Chadwick; Otenko, 2002).	38
Quadro 4 - Comparação entre as plataformas de grade estudadas.....	56
Quadro 5 - Privilégios utilizados para realização de testes.	115
Quadro 6 - Equipamentos utilizados nos testes práticos.....	121
Quadro 7 – Tempo para detecção de ataques.	125

LISTA DE ABREVIATURAS E SIGLAS

AA	-	Autoridade Atribuidora
AC	-	Autoridade Certificadora
ACL	-	<i>Access Control List</i>
API	-	<i>Application Programming Interface</i>
ASL	-	<i>Authorization Specification Language</i>
DAC	-	<i>Discretionary Access Control</i>
DAI	-	<i>Data Access and Integration</i>
CAS	-	<i>Community Authorization Service</i>
GT4	-	<i>Globus ToolKit version 4</i>
ICP	-	Infra-estrutura de Chave Pública
IDMF	-	<i>Intrusion Detection Message Exchange Format</i>
IDS	-	<i>Intrusion Detect System</i>
IEFT	-	<i>Internet Engineering Task Force</i>
IPS	-	<i>Intrusion Prevention Systems</i>
IP	-	<i>Internet Protocol</i>
LESS	-	Linguagem para Expressar Serviços de Segurança
JMPE	-	<i>Java Multi Policy Environment</i>
JVM	-	<i>Java Virtual Machine</i>
LRTL	-	<i>Legion Runtime Library</i>
MLS	-	<i>Multi Level Secure System</i>
MPMS	-	<i>Multi Policy Manager Service</i>
OGF	-	<i>Open Grid Forum</i>
OGSA	-	<i>Open Grid Services Architecture</i>
PAP	-	Ponto de Administração de Política
PEP	-	Ponto de Efetivação de Política
PDP	-	Ponto de Decisão de Política
PIP	-	Ponto de Informativo de Política
PMI	-	<i>Privilege Management Infrastructure</i>
SAML	-	<i>Security Assertion Markup Language</i>
SOA	-	<i>Source of Authority</i>

<i>SOAP</i>	-	<i>Simple Object Access Protocol</i>
<i>SPKI</i>	-	<i>Simple Public Key Infrastructure</i>
<i>SSH</i>	-	<i>Secure Shell</i>
<i>SSL</i>	-	<i>Secure Socket Layer</i>
<i>SSO</i>	-	<i>Single Sign On</i>
<i>RBAC</i>	-	<i>Role Based Access Controls</i>
<i>RDA</i>	-	<i>Receptor de Detecção de Ataques</i>
<i>RLF</i>	-	<i>Replica Location Service</i>
<i>RSL</i>	-	<i>Resource Specification Language</i>
<i>TCP</i>	-	<i>Transmission Control Protocol</i>
<i>VO</i>	-	<i>Virtual Organization</i>
<i>VPN</i>	-	<i>Virtual Private Network</i>
<i>WS GRAM</i>	-	<i>Web Service Grid Resource Allocation and Management</i>
<i>XACML</i>	-	<i>eXtensible Access Control Markup Language</i>
<i>XML</i>	-	<i>eXtensible Markup Language</i>

SUMÁRIO

1 INTRODUÇÃO	16
1.1 Justificativa e Motivação	18
1.2 Objetivos	19
1.3 Metodologia	20
1.3.1 Estrutura do trabalho	20
2 TECNOLOGIAS DE SEGURANÇA INTEROPERÁVEIS	22
2.1 Aspectos de Segurança em Aplicações Distribuídas	22
2.2 Políticas de Segurança	24
2.2.1 Modelos de Controle de Acesso	24
2.2.2 Lógica Formal para Representar Políticas de Segurança	26
2.2.3 Linguagem para Expressar Serviços de Segurança (LESS)	32
2.2.4 <i>XACML</i>	33
2.3 Efetivando Políticas de Segurança	35
2.4 Infra-Estruturas para Distribuição de Políticas	36
2.4.1 Certificados X.509	36
2.4.2 <i>PMI</i>	37
2.4.3 <i>SPKI</i>	39
2.4.4 MyProxy	39
2.4.5 Comparando as Tecnologias para Distribuição de Políticas	40
2.5 Conclusões	41
3 SEGURANÇA EM GRADES COMPUTACIONAIS	42

	17
3.1 Grade Computacional	42
3.2 Requisitos de Segurança para Grades Computacionais	44
3.3 Modelo de segurança OGSA	47
3.4 Plataformas de Segurança em Ambientes de Grade Computacional	48
3.4.1 <i>Globus Toolkit 4</i>	49
3.4.2 <i>SGE</i>	51
3.4.3 <i>Legion</i>	51
3.4.4 <i>OurGrid</i>	52
3.4.5 <i>InteGrade</i>	52
3.5 Analisando as Soluções de Segurança para Grades Computacionais	54
3.6 Soluções de Controle de Acesso em Grades Computacionais – Trabalhos Relacionados	57
3.6.1 Infra-estruturas de Controle de Acesso em Serviços de Rede	57
3.6.2 Controle de Acesso em Instanciação Remota de Tarefas	61
3.6.3 Análise das Soluções de Sistema de Controle de Acesso para Grades Computacionais	64
3.7 Conclusões	65
4 MODELO GRIDMULTIPOLICY	67
4.1 Pré-requisitos e Visão Geral	67
4.2 Modelo de Gerenciamento de Políticas de Múltiplos Formatos	70
4.2.1 Escopo de utilização de políticas	70
4.2.2 Informações para a Contextualização de Políticas	71
4.2.3 Integração com Mecanismos Nativos	71
4.2.4 Certificados de Políticas	73
4.2.5 Suporte ao Princípio de Privilégio Mínimo	73
4.3 Modelo de Efetivação de Políticas	74

	18
4.4 Conclusões	75
5 GRIDMULTIPOLICY – ARQUITETURA E IMPLEMENTAÇÃO	77
5.1 Arquitetura GridMultiPolicy - Visão Geral	77
5.2 JMPE	78
5.2.1 Estabelecimento dos PDPs	79
5.2.2 Estabelecimento dos PEPs	80
5.2.3 Integrando Níveis de Granularidade	88
5.3 MPMS	89
5.3.1 MPMS Visão Geral	89
5.3.2 Extensão da Linguagem <i>XACML</i>	90
5.3.3 Gerenciamento de Privilégios	92
5.3.4 Distribuição de Políticas	95
5.4 Integrando GridMultiPolicy ao GT4	96
5.4.1 Integrando GridMultiPolicy ao <i>WS Gram</i>	96
5.4.2 Integrando GridMultiPolicy ao <i>Java WS Core</i>	96
5.5 Conclusões	98
6 CENÁRIOS DE USO E ESTUDO DE CASO	99
6.1 Cenário C1 – Instanciação de Tarefas e Serviços em Sítios com Mecanismos de Proteção	99
6.2 Cenário C2 – Uso de Serviços de Bancos de Dados por Aplicações da Grade Computacional	101
6.3 Cenário C3 – Laboratórios Remotos	101
6.4 Cenário C4 - Controle de Acesso a Serviços de rede da Grade Computacional	103
6.5 Implementando Políticas	104
6.5.1 Política LeastPrivilege	105

	19
6.5.2 Política IDS_Control	106
6.5.3 Política DBModel	108
6.6 Estudo de caso	110
6.6.1 Política lab_Control	113
6.6.2 Teste Operacional	115
6.7 Conclusões	117
7 ANÁLISE DE DESEMPENHO	119
7.1 Ambiente de Testes	120
7.2 Avaliação do Impacto de Desempenho na Instanciação e Execução de Tarefas Remotas	121
7.3 Avaliação do Impacto na Instanciação e Oferecimento de Serviços	126
7.4 Conclusões	128
8 CONCLUSÕES, LIMITAÇÕES E FUTUROS TRABALHOS	130
8.1 Conclusões	130
8.2 Limitações	132
8.3 Futuros Trabalhos	133
REFERÊNCIAS BIBLIOGRÁFICAS	134

1 Introdução

O termo “grade computacional” faz referência a uma classe de sistemas distribuídos que permite a associação e integração de múltiplos domínios independentes em organizações virtuais (*Virtual Organizations – VO*) (FOSTER, 2005; BAVIER et al., 2004). Para muitos, as grades computacionais serão a base tecnológica para o oferecimento de infra-estruturas mais potentes e flexíveis no século XXI, pois permitirão uma maior eficiência no estabelecimento e gerenciamento de sistemas. Contudo, para viabilizar previsões como esta, diversos desafios precisam ser transpostos, entre os quais a realização de um serviço de controle de acesso eficiente e integrado sobre um ambiente distribuído e heterogêneo (WELCH et al., 2003). Seu correto oferecimento deve contribuir para o uso de um modelo de maior integração, sem agravar os riscos dos sistemas envolvidos.

Corporações dependem, cada vez mais, de informações contidas em computadores para que seus sistemas possam executar suas atividades corretamente. Ataques que indisponibilizem, interceptem, modifiquem e/ou fabriquem informações podem acarretar enormes e graves prejuízos para corporações e indivíduos. Desta maneira, a segurança é um requisito fundamental para que as tecnologias de grade computacional, hoje utilizadas quase que exclusivamente em ambientes acadêmicos, possam ser adotadas em um contexto mais amplo.

A segurança em redes é um tema já com certo grau de maturidade, evidenciado pelo grande número de mecanismos e tecnologias — dentre as quais podemos citar *Firewalls*, os Sistemas de Detecção de Intrusão (*Intrusion Detection Systems – IDS*) e as Redes Privadas Virtuais (*Virtual Private Network – VPN*). Assim, em vez de substituir, a grade computacional deve fazer uso das infra-estruturas existentes, aproveitando o conhecimento já estabelecido e reduzindo os custos para sua implementação (WELCH, 2003).

Um típico *ambiente* de grade computacional, por um lado, oferece um ambiente unificado e integrado sobre múltiplos domínios, no qual os recursos são oferecidos e gerenciados de forma dinâmica, sem a intervenção direta de

administradores. Por outro lado, deve permitir que cada domínio ou usuário desenvolva e gerencie seus recursos de forma autônoma (FOSTER, 2005) (ABRZYSKI, SCHOPF, 2003). Desta maneira, um serviço coerente de autorização para grades computacionais deve oferecer suporte a múltiplas políticas, exercendo um controle unificado sobre os recursos disponibilizados, além de permitir que domínios e usuários estabeleçam, de forma autônoma, os mecanismos e as políticas para controlar seus recursos.

Para demonstrar a importância dos requisitos descritos, tomemos a título de exemplo uma VO que reúna as seguintes necessidades de segurança e situações:

- Um administrador deseja aumentar a segurança da VO. Nesse intuito, implementa uma política baseada no princípio de privilégio mínimo, que deve ser efetivada em todos os sítios associados;
- Um usuário, ao armazenar e disponibilizar suas informações e aplicações nos sítios dessa VO, deseja impor suas próprias regras para controlar o acesso de outros usuários a seus recursos;
- Em um sítio de uma VO, o administrador deseja integrar um mecanismo IDS com o ambiente de execução, a fim de detectar e bloquear aplicações maliciosas provenientes de outros domínios. Tal funcionalidade teria grande valor para diminuir os riscos associados ao serviço da grade computacional de instanciamento remoto de tarefas, que pode representar uma grande vulnerabilidade aos sistemas (KEAHEY; RIPEANU; DOERING, 2004).
- Uma instituição de pesquisa, por meio da grade computacional, disponibiliza seus equipamentos para a realização de experimentos remotos. Para evitar excessos na utilização dos recursos envolvidos, necessita de meios para efetivar suas políticas de segurança.

No intuito de contribuir para a evolução dos sistemas de grade computacional, o presente trabalho desenvolveu o sistema GridMultiPolicy como proposta para um novo modelo de controle de acesso, que faça frente aos requisitos de segurança envolvidos. O modelo proposto tem como característica principal a flexibilidade que lhe permite gerenciar e efetivar políticas de segurança de múltiplos modelos, formatos e granularidades.

1.1 Justificativa e Motivação

Apesar dos avanços obtidos em trabalhos recentes, as soluções encontradas de controle de acesso em grades computacionais apresentam limitações em pelo menos dois aspectos.

Em primeiro lugar, apesar dos esforços para incorporar múltiplas fontes ao processo de autorização, falta uma solução que faça frente às necessidades das grades computacionais, no que diz respeito ao gerenciamento e à efetivação de múltiplas políticas de segurança desenvolvidas de forma independente¹.

Em segundo lugar, os trabalhos produzidos seguem duas orientações distintas: uma voltada para infra-estruturas distribuídas e flexíveis para exercer controle de acesso nos serviços de rede da grade computacional; outra, menos flexível, voltada para configuração de ambientes de execução, a fim de limitar as ações das tarefas instanciadas remotamente.

Uma solução mais abrangente ofereceria flexibilidade não só para controlar o acesso de usuários a serviços de rede, mas também para controlar as operações que tarefas e serviços venham a realizar no ambiente de execução. Tal capacidade seria de grande importância, pois, em boa parte dos acessos a recursos, o ambiente de execução é a única instância de controle, e a grade computacional permite a instanciamento remoto de tarefas e serviços que podem ser desenvolvidos de forma maliciosa ou ter falhas de segurança.

Esta tese pretende abordar esses pontos, apresentando um modelo de autorização mais adequado às necessidades das grades computacionais, que gerencie e efetive múltiplas políticas independentes e possua a abrangência necessária para controlar tanto o oferecimento de serviços de rede, como as ações no ambiente de execução realizadas por aplicações (tarefas ou serviços).

O alto custo para a execução de pesquisas científicas e a desigualdade econômica entre países sugerem que o correto emprego de tecnologias colaborativas, como a grade computacional, pode ser um entre os muitos fatores que possibilitem a um país periférico como o Brasil diminuir seu atraso tecnológico. A

¹ O termo “independente” é utilizado no sentido caracterizar políticas desenvolvidas de forma autônoma com sua própria linguagem, formato, granularidades e/ou mecanismos de efetivação.

possibilidade de entender e contribuir com uma disciplina que possa favorecer o desenvolvimento nacional foi um fator motivacional fundamental para a realização da presente tese. Desta maneira, buscou-se também oferecer um retorno apropriado ao investimento público realizado na concepção da presente pesquisa.

1.2 Objetivos

Este trabalho tem como objetivo principal propor um novo modelo de controle de acesso que ofereça respostas aos desafios de segurança das grades computacionais. O modelo proposto deve ser capaz de oferecer um serviço de segurança integrado sobre ambientes heterogêneos e com necessidades distintas, devendo se adaptar dinamicamente a cada contexto de segurança, de forma a oferecer suporte aos requisitos previstos para proteger os recursos envolvidos.

Para poder cumprir seu objetivo principal, o modelo de segurança proposto deve atingir os objetivos secundários propostos para esta tese, que é o oferecimento de respostas adequadas aos seguintes desafios e requisitos para grades computacionais:

- a. Permitir a integração de políticas de domínios autônomos. Dentro de um contexto heterogêneo como uma grade computacional, um modelo ou formato de política não preenche todos os requisitos necessários (DAMIANI; VIMERCATI; SAMARATI, 2005), (COETZEE; ELOFF, 2003). Deste modo, deve-se permitir que administradores, usuários e programadores da VO possuam liberdade para estabelecer suas próprias políticas de forma autônoma;
- b. Oferecer serviços de segurança de forma integrada com os mecanismos pré-existent. O custo associado à substituição das infra-estruturas de segurança preexistentes dificulta em muito a implementação de tecnologias de grades. Assim, tal requisito visa, principalmente, a redução de custos (KEAHEY; et al, 2004);
- c. Oferecer meios para o estabelecimento flexível de relações de confiança, no intuito de coordenar os recursos de uma VO que tem características dinâmicas (KEAHEY; et al; 2004);
- d. Efetivar políticas de suporte ao princípio de privilégio mínimo. Este princípio define que uma tarefa deve ter os privilégios para sua correta execução, não

mais que isso. Essa característica limita os possíveis danos a serem causados pelo emprego de programas maliciosos ou com falhas de segurança (KEAHEY; et al; 2004);

- e. Oferecer uma solução abrangente, que possa ser utilizada para exercer controle, tanto no oferecimento de serviços, como no ambiente de execução das aplicações;
- f. Integrar políticas de granularidades distintas, permitindo a efetivação conjunta de políticas que tenham recursos-alvos em comum, porém, que atuem em níveis de controle diferentes².
- g. Promover uma solução que não exija a reformulação das aplicações, não criando, desta maneira, problemas de interoperabilidade com as aplicações legadas e/ou com outras tecnologias de grade computacional.

1.3 Metodologia

O ambiente de execução e os componentes do sistema de autorização GridMultiPolicy, que é uma materialização do modelo proposto por esta tese, foram realizados tendo como base a plataforma e a linguagem de programação Java. A implementação oferecida foi concebida para ser utilizada de forma integrada a diferentes plataformas de grade computacional.

Para a realização dos testes, o GridMultiPolicy foi integrado à plataforma Globus Toolkit 4.0 (GT4) (FOSTER, 2005).

1.3.1 Estrutura do trabalho

Esta tese é composta de oito capítulos, estruturados de forma a apresentar primeiramente um panorama dos principais desafios e tecnologias correlacionados com a segurança em grades computacionais e, posteriormente, o modelo de controle de acesso proposto.

² Os conceitos de nível de granularidade utilizados nesse trabalho são definidos com maior propriedade na Seção 4.1.

O Capítulo 2 descreve as principais tecnologias de controle de acesso em sistemas distribuídos. Em seguida, o Capítulo 3 oferece, de forma sucinta, as mais avançadas tecnologias de grade computacionais, apontando desafios de segurança ainda não solucionados, os quais a presente tese se ocupa em transpor.

Os Capítulos 4 e 5 descrevem o modelo, a arquitetura e a implementação do sistema de segurança apresentado por esta tese. Ao final, além de uma visão ampla do GridMultiPolicy, o leitor poderá ter a concepção dos processos e técnicas empregadas na sua implementação.

O Capítulo 6 apresenta cenários de usos, exemplos de políticas e um estudo de caso que, em conjunto, permitem ter uma visão da capacidade do sistema proposto em oferecer respostas às necessidades de segurança presentes nas grades computacionais.

No intuito de validar o modelo de controle de acesso proposto, o Capítulo 7 contém os resultados de testes de desempenho realizados em diferentes condições de uso do GridMultiPolicy.

2 Tecnologias de Segurança Interoperáveis

Com a difusão de sistemas computacionais por todas as áreas do conhecimento humano, a utilização de redes locais e globais passou a ter um papel fundamental, oferecendo suporte a diversos serviços. Tais serviços permitem gerar aplicações que envolvem transações financeiras, bancos virtuais, vendas, busca por documentos, propagandas utilizadas em grande escala, só para citar alguns exemplos.

Dessa forma, dado o papel fundamental que as redes ocupam hoje em dia, somado ao grande número de ataques computacionais, a área de segurança passou a ser amplamente estudada nos meios acadêmicos e comerciais, nos quais se encontra uma grande variedade de produtos, como *Firewalls*, Rede Privada Virtual (*Virtual Private Network - VPN*) e Sistema de Detecção de Intrusão (*Intrusion System Detect - IDS*) etc. Nesse sentido, o estudo da segurança em ambientes computacionais passou a ser bastante amplo, abrangendo as áreas de sistemas operacionais, tecnologias de redes, desenvolvimento de softwares, engenharia social, dentre outras. Contudo, diante da diversidade gerada, a integração tecnológica passou a ser um desafio a ser vencido para a criação de sistemas distribuídos interoperáveis.

Neste capítulo, serão apresentados alguns conceitos e tecnologias de segurança em sistemas distribuídos, estudados para a formulação da presente tese.

2.1 Aspectos de Segurança em Aplicações Distribuídas

Além de oferecer forte proteção, espera-se que os modernos sistemas de segurança distribuídos sejam flexíveis o suficiente para promover a interoperabilidade entre domínios. Modelos de segurança baseados em políticas, ao separarem as decisões de comportamento da implementação, permitem que, em contexto de execução, os mecanismos possam ser estabelecidos e configurados de forma flexível e dinâmica. Nesse sentido, MARSHALL, JAJODIA E PODELL (1995) definem que o oferecimento de serviços de segurança em ambientes computacionais é um processo que envolve três conceitos fundamentais:

1. Políticas de segurança: leis, regras e práticas que regulam como uma organização gerencia, protege e distribui informações;
2. Funcionalidades: mecanismos internos para fazer cumprir políticas de segurança;
3. Garantias sobre a eficácia: provas a respeito da capacidade dos mecanismos em fazer cumprir as políticas estabelecidas.

O trabalho CORBASEC (CORBASEC, 2007) oferece uma especificação sobre sistemas computacionais distribuídos seguros, apontando a necessidade dos seguintes serviços:

- **Autenticação:** funcionalidade que, por meio de interações, permite que uma entidade certifique-se da identidade de outra parceira de comunicação. Há uma infinidade de sistemas de autenticação com diferentes custos computacionais e níveis de segurança, que fazem uso de inúmeros métodos e tecnologias, como algoritmos de criptografia simétrica e assimétrica, senha secreta, parâmetros biométricos etc.;
- **Controle de acesso:** funcionalidade responsável por autorizar ou negar o acesso a determinado recurso do sistema para uma entidade. Seu funcionamento é baseado em modelos de políticas em que privilégios e permissões de acesso são definidos para grupos, papéis e usuários, de forma a controlar o acesso aos arquivos, diretórios, máquinas, serviços e outros;
- **Segurança na comunicação (integridade e confidencialidade):** funcionalidade responsável por garantir a confidencialidade e a integridade dentro de um contexto de comunicação, no qual somente entidades participantes e autorizadas poderão acessar, modificar e redigir mensagens, de modo que qualquer alteração no conteúdo desta informação seja detectada. Com esse intuito são utilizadas duas técnicas diferentes: a integridade garantida por funções espalhamento (*hash*) e a privacidade garantida por algoritmos de criptografia simétrica ou assimétrica. Entre os diversos algoritmos de espalhamento podemos citar: *MD4 (Message Digest 4)*, *MD5 (Message Digest 5)* e *SHA-1 (Secure Hash Algorithm-1)* (STALLINGS, 1998); de criptografia simétrica *DES (Data Encryption Standard)*, *RC4 (Rivest Cipher 4)*, *RC5 (Rivest Cipher 5)* e *IDEA (International Data Encryption Algorithm)* (STALLINGS, 1998).

- **Auditoria de segurança:** define uma série de eventos e contextos que devem gerar notificações ou armazenamento de informações;
- **Irretratabilidade:** a funcionalidade de irretratabilidade permite responsabilizar os usuários pelos seus atos, por meio do armazenamento de provas irrefutáveis de ações específicas. A infra-estrutura de chaves públicas (STEVE; CARLISLE, 2002) possibilita a realização de provas para a irretratabilidade, utilizando como meio a assinatura digital de documentos por parte de entidades certificadas, que oferece garantias sobre a identidade do redator e do conteúdo do documento;
- **Administração:** é necessária para a configuração e administração de usuários e políticas de segurança, além de prover relatórios e notificações relacionados à segurança do sistema. Temos hoje em dia uma infinidade de ferramentas administrativas, como consoles administrativos, leitores de *logs*, varredores de vulnerabilidades etc.

2.2 Políticas de Segurança

As políticas de segurança fornecem as leis, as regras e as práticas que regulam como uma organização gerencia, protege e distribui informações, separando a tomada de decisão da implementação, características estas que atribuem uma maior flexibilidade aos sistemas baseados em políticas.

Nos sistemas computacionais, as políticas são escritas segundo uma linguagem específica para, posteriormente, serem interpretadas e efetivadas por mecanismos. Assim, a integração de sistemas heterogêneos passa por interoperar os diferentes modelos de políticas envolvidos.

Esta Seção descreve os principais modelos de políticas existentes. Posteriormente, serão apresentadas as lógicas formais que serviram de base para a criação dos padrões *SAML* (SAML, 2005) e *XACML* (XACML, 2006) para a interoperabilidade de políticas de controle de acesso.

2.2.1 Modelos de Controle de Acesso

O modelo de controle de acesso constitui uma representação formal das

regras para tomada de decisões, isto é, negar ou autorizar requisições de ações sobre recursos e dados de um sistema (SANDHU; SAMARATI, 1994). Tal formalização se mostra útil na realização de mecanismos de controle de acesso e seu entendimento.

Os modelos de controle de acesso podem ser classificados em três grupos (SANDHU; SAMARATI, 1994):

- a) **Controle de Acesso Discriminante (*Discretionary Access Control - DAC*)**, no qual as decisões de acesso são baseadas na identidade do solicitante e em regras que indicam quais entidades devem ou não ter acesso;
- b) **Sistema de Segurança com Diversos Níveis (*Multi Level Secure System - MLS*)**, no qual as decisões de acesso são baseadas em rótulos classificatórios de objetos;
- c) **Controle de Acesso Baseado em Papéis (*Role Based Access Controls - RBAC*)** baseia-se em papéis assumidos por usuários para realizar as decisões de acesso.

DAC

O mais tradicional esquema de autorização é o *DAC* no qual os direitos de acesso são gerenciados pelo administrador e discriminados no sistema por meio de uma Lista de Controle de Acesso (*Access Control List - ACL*) específica para um determinado recurso. Dentro de uma *Privilege Management Infrastructure (PMI)* os atributos de acesso ficam contidos em credenciais X.509 (ADAMS; FARRELL, 1999) que contêm atributos para um ou mais recursos.

A utilização do modelo *DAC* tem como principal vantagem a flexibilidade, que permite restringir e garantir acesso para usuários de forma individual. Porém, sua utilização em grandes domínios torna o gerenciamento difícil, dado o número de usuários e de recursos a serem administrados individualmente.

MLS

O Sistema de Segurança com Diversos Níveis (*Multi Level Secure System -*

MLS) (SANDHU; SAMARATI, 1994) é muito popular em ambientes militares. Conta com uma estrutura classificatória hierárquica, que atribui rótulos de controle de acesso aos recursos e aos usuários do sistema, utilizando listas para especificar que tipo de acesso o usuário tem para cada rótulo. Usualmente, os seguintes termos de rótulos são utilizados para representar níveis crescentes de privilégio: sem classificação, restrito, confidencial, secreto e supersecreto.

RBAC

O Controle de Acesso Baseado em Papéis (*Role Based Access Controls – RBAC*) (SANDHU; SAMARATI, 1994) utiliza a atribuição de privilégios a papéis que são previstos de acordo com a estrutura funcional da instituição, tais como: secretária, analista, gerente, presidente. Todos os usuários do sistema são associados a um ou mais papéis e assumem os privilégios dos mesmos.

O modelo RBAC pode contar ainda com uma organização hierárquica, de maneira que os papéis superiores possuam todos os direitos dos inferiores. Dessa forma, a atribuição de um direito a um papel implica que todos os papéis hierarquicamente superiores possuirão o mesmo direito.

O grande atributo do RBAC é a facilidade administrativa e a representação compacta dos atributos de acesso.

2.2.2 Lógica Formal para Representar Políticas de Segurança

A questão mais importante relativa à integração de serviços de segurança sobre redes heterogêneas é a interoperabilidade entre as diferentes tecnologias, cada qual com seus mecanismos, modelos e parâmetros. O primeiro passo para prover tal integração é fornecer meios para que as diferentes regras e estruturas de dados utilizadas possam ser representadas por meio de uma única sintaxe. Esta seção apresenta algumas linguagens formais para a representação de diferentes modelos de políticas.

Linguagem para Especificação de Autorização (*Authorization Specification Language - ASL*)

A linguagem *ASL* (JAJODIA; SUBRAHMANIAN, 1997) foi uma das primeiras tentativas de representação de múltiplas políticas por meio de uma linguagem lógica.

Definição: a política de controle de acesso mapeia tuplas contendo {**o**, **s**, **p**, **a**}, objeto, sujeito, papel e ação dentro do conjunto {**+a** (autorizado), **-a** (negado)}, como a seguinte representação: $f\{\mathbf{o}, \mathbf{s}, \mathbf{p}, \mathbf{a}\} \rightarrow \{+\mathbf{a}, -\mathbf{a}\}$.

A linguagem a ser descrita conta com o seguinte alfabeto:

- ✓ **Constantes:** todos os membros utilizados pela linguagem pertencem a: {**OBJ**, **T**, **U**, **G**, **P**, **A**, **NA**}, sendo **OBJ** o conjunto dos objetos, **T** o conjunto dos tipos de objetos, **U** o conjunto dos usuários, **G** o conjunto dos grupos, **P** o conjunto dos papéis, **A** o conjunto das ações e **NA** = {**+a**, **-a**}
- ✓ **Variáveis:** as variáveis: **Vo**, **Vt**, **Vu**, **Vg**, **Vp**, **Va** e **Vna** pertencem respectivamente a **OBJ**, **T**, **U**, **G**, **P**, **A** e **NA**.
- ✓ **Predicados:** são utilizados os seguintes predicados:
 1. *pode* - representa uma autorização explícita de um sujeito para realizar uma determinada ação em um objeto;
 2. *derive_pode* - representa uma regra para derivação de autorização (*pode*), utilizada para definir a transferência de atributos de grupos e papéis para seus membros;
 3. *faça* - representa a regra para a determinação de autorização para sujeitos, sendo sua utilização necessária para a resolução de conflitos provenientes da derivação de autorização;
 4. *permitido* - representa a regra responsável por autorizar ou negar a requisição de uma ação;
 5. *ativar* - representa a ligação entre um papel e um sujeito;
 6. *atribui* - representa a ligação entre um sujeito **s** e um grupo **g**: *atribui*{**s**, **g**};
 7. *ativo* - verifica se um sujeito **s** pertence a um grupo **g** ou está ativo em um papel **p**: *ativo* {**s**, {**g**, **p**}} $\rightarrow$ {verdadeiro, falso};

8. *tipode* - verifica a relação entre um objeto e um tipo: *tipode* {obj, t}
 $\rightarrow \{\text{verdadeiro, falso}\}.$

Seguem agora as definições das regras que podem ser representadas por nossa ASL:

Pode: representa uma regra para autorização explícita para que sujeitos possam realizar ações “a” sobre objetos “o” e possui a seguinte sintaxe: *pode* (o, s, <sinal>a) $\leftarrow \{L_1 \ \& \ L_2 \ ..L_n\}$

A expressão L_1 representa condições a serem respeitadas para a validade da regra e pode ser realizada a partir dos predicados apresentados e parâmetros do sistema.

Tomemos como exemplos as seguintes expressões:

1. *Pode*(arquivo_1, Pedro, +ler).
2. *Pode*(o, contador, +escrever). $\leftarrow tipode(o, planilha)$
3. *Pode*(arquivo_2, secretária, -escrever).

A primeira expressão representa uma permissão para o usuário “Pedro” “ler” o objeto “arquivo_1”. A segunda expressão representa uma permissão para o papel “contador” “escrever” objetos que sejam do tipo “planilha”. A última expressão representa uma negativa de “escrever” no “Arquivo_2” para o grupo “secretária”.

Derive_pode: representa uma regra para derivação de parâmetros de autorização, possuindo a seguinte sintaxe: *derive_pode*(o,s, <sinal>a) $\leftarrow \{ L_1 \ \& \ L_2 \ ..L_n\}.$

A expressão L_1 representa condições a serem respeitadas para a validade da regra e pode ser realizada a partir dos predicados apresentados e dos parâmetros do sistema.

Tomemos como exemplos, as seguintes expressões:

1. *Derive_pode*(o,s,+a) $\leftarrow \{ pode(o, s', +a) \ \& \ ativo(s, s') \}$
2. *Derive_pode*(o,s,+ler) $\leftarrow (pode(o, s', +ler) \ \& \ ativo(s, s')) \ \& \ (pode(o,s,+Escrever) = \phi)$

A primeira expressão realiza a derivação de autorizações positivas “+a” sobre objetos “o” de grupos e papéis “s” para todos os seus membros “s’”. De acordo com a segunda expressão, uma derivação positiva de “ler” só pode ocorrer se o sujeito não possuir uma permissão explícita positiva para “escrever”.

Faça: representa uma regra para resolução de autorização para sujeitos,

possuindo a seguinte sintaxe: $fa\grave{c}a(o,s, <sinal>a) \leftarrow \{L1 \& L2 \dots Ln\}$

Sua utiliza\c{c}o\~{a}o permite que regras para a atribuic\~{a}o de par\~{a}metros de autoriza\c{c}o\~{a}o sejam representadas, tornando-se assim \u00fasil para resolu\c{c}o\~{a}o de conflitos provenientes da deriva\c{c}o\~{a}o de privil\~{e}gios.

Tomemos como exemplos as seguintes express\~{o}es:

$$fa\grave{c}a(o,s, +a) \leftarrow ((derive_pode(o,s, +a) \& (pode(o,s, -a) = \phi) \oplus (pode(o,s, +a)))$$

$$fa\grave{c}a(o,s, -a) \leftarrow (derive_pode(o,s, -a) \oplus (pode(o,s, -a)))$$

No primeiro exemplo, uma autoriza\c{c}o\~{a}o positiva de uma a\c{c}o\~{a}o \u00e9 relacionada a um sujeito, caso possua uma positiva direta *pode* ou se possuir uma deriva\c{c}o\~{a}o positiva e n\~{a}o possuir uma negativa direta. No segundo exemplo, uma negativa \u00e9 relacionada ao sujeito, se ele possuir uma negativa direta ou derivada.

Permitida: representa uma regra para negar ou autorizar requisic\~{o}es de uma a\c{c}o\~{a}o por um sujeito em objetos, possuindo a seguinte sintaxe: $permitida(o,s, +a) \leftarrow \{L1 \& L2 \dots Ln\}$.

Exemplos: tomemos como exemplos as seguintes express\~{o}es:

$$permitida(o,s, +a) \leftarrow fa\grave{c}a(o,s, +a).$$

$$permitida(o,s, +a) \leftarrow (fa\grave{c}a(o,s, -a) = \phi).$$

No primeiro exemplo, uma autoriza\c{c}o\~{a}o \u00e9 fornecida somente quando existe uma express\~{a}o positiva para sua realiza\c{c}o\~{a}o. No segundo, uma autoriza\c{c}o\~{a}o \u00e9 fornecida somente se n\~{a}o possui uma express\~{a}o negativa para sua realiza\c{c}o\~{a}o.

Ativar: representa uma regra para ativa\c{c}o\~{a}o de pap\~{e}is por usu\~{a}rios, possuindo a seguinte sintaxe: $ativar(u,p) \leftarrow \{L1 \& L2 \dots Ln\}$

A express\~{a}o L_1 , realizada a partir de par\~{a}metros do sistema, fornece as regras para o usu\~{a}rio **s** ativar o papel **p**.

Tomemos como exemplos as seguintes express\~{o}es:

$$Ativar(Pedro, Gerente) \leftarrow (IP_des \geq 10.0.161.32 \& IP_des \leq 10.0.161.62)$$

$$Ativar(Jo\~{a}o, T\acute{e}cnico) \leftarrow (hora < 20:00)$$

No primeiro exemplo, Pedro ativa o papel de gerente somente quando requisita recursos de m\~{a}quinas com endere\c{c}o *IP* de 10.0.161.32 at\~{e} 10.0.161.62. No segundo exemplo, Jo\~{a}o ativa o papel de t\acute{e}cnico em hor\~{a}rios inferiores a 20h00.

O modelo hier\~{a}rquico para grupos e pap\~{e}is prev\~{e} privil\~{e}gios transferidos de um sujeito hierarquicamente acima para os sujeitos hierarquicamente abaixo. Assim, al\~{e}m dos privil\~{e}gios diretamente relacionados a si, um sujeito pode derivar privil\~{e}gios

dos grupos e papéis aos quais pertence. Como uma determinada ação sobre um objeto não pode ser negada e autorizada para uma mesma entidade, a derivação de privilégios pode gerar conflitos.

Os seguintes exemplos ilustram tal situação:

- Um usuário possui uma autorização direta para ler um arquivo e é membro de um grupo que possui uma negativa para ler o mesmo arquivo. Em uma derivação direta de privilégio, tal usuário possuiria duas permissões contrárias.
- Um usuário é membro de dois grupos: um possui uma autorização positiva para uma ação sobre um arquivo, enquanto o outro grupo possui uma autorização negativa para a mesma ação. Nesse exemplo, a derivação direta também geraria conflito.

O processo de delegação de privilégios tem o objetivo de impedir a ambigüidade de privilégios, resolvendo conflitos de propagação e ponderando, nesse sentido, sobre dois aspectos da política de autorização descritos a seguir:

- ✓ Derivação: regula o processo de transferência de autorização ao longo da hierarquia;
- ✓ Resolução de conflito: determina qual autorização deve prevalecer em caso de conflitos.

A seguir, serão discutidas as diferentes escolhas a serem feitas sobre tais aspectos.

A política de derivação de privilégios define regras para a derivação de autorização ao longo de uma hierarquia de grupos e papéis até as entidades finais. Três diferentes modelos de propagação são mostrados a seguir:

- ✓ Sem sobreposição: nesse modelo, todos os privilégios são propagados, deixando a decisão para a resolução de conflitos;
- ✓ Sobreposição do subsujeito: a autorização especificada para o sujeito hierarquicamente abaixo é adotada em caso de conflito;
- ✓ Sobreposição do supersujeito: a autorização especificada para o sujeito hierarquicamente acima é adotada em caso de conflito.

Utilizando a linguagem *ASL*, o Quadro 1 demonstra a representação dos modelos de propagação e resolução de conflitos acima descritos.

Modelo	Expressão
Sem sobreposição	$derive_pode(o, s, a) \leftarrow pode(o, s, a) \& ativo(s, s')$
Sobreposição do sub-sujeito	$derive_pode(o, s, a) \leftarrow pode(o, s, a) \& ativo(s, s') \& (\emptyset = (pode(o, s, -a)))$
Sobreposição do supersujeito	$derive_pode(o, s, a) \leftarrow pode(o, s, a) \& ativo(s, s') \& (\emptyset = (pode(o, s'', -a) \& ativo(s', s'') \& s' < s''))$

Quadro 1 - Exemplo de utilização da linguagem lógica para representar modelos de derivação de privilégios.

A resolução de conflito tem como objetivo impedir que uma decisão de acesso ao sistema possa ser simultaneamente autorizada e negada. Os seguintes modelos de resoluções de conflitos são propostos:

1. A recusa tem preferência: em caso de conflito a decisão é negada;
2. O acesso tem preferência: em caso de conflito o acesso é permitido;
3. Nada é sobreposto: em caso de conflito o sistema age como se nada fosse especificado para a permissão requerida.

Utilizando a linguagem *ASL*, o Quadro 2 demonstra a representação dos modelos para resolução de conflitos.

Modelo	Expressão
Recusa tem preferência	$\text{faça}(\mathbf{o}, \mathbf{s}, +\mathbf{a}) \leftarrow \text{derive_pode}(\mathbf{o}, \mathbf{s}, +\mathbf{a}) \&$ $(\emptyset = \text{derive_pode}(\mathbf{o}, \mathbf{s}, -\mathbf{a}))$ $\text{faça}(\mathbf{o}, \mathbf{s}, -\mathbf{a}) \leftarrow \text{derive_pode}(\mathbf{o}, \mathbf{s}, -\mathbf{a})$
Acesso tem preferência	$\text{faça}(\mathbf{o}, \mathbf{s}, +\mathbf{a}) \leftarrow \text{derive_pode}(\mathbf{o}, \mathbf{s}, +\mathbf{a})$ $\text{faça}(\mathbf{o}, \mathbf{s}, -\mathbf{a}) \leftarrow \text{derive_pode}(\mathbf{o}, \mathbf{s}, -\mathbf{a}) \&$ $(\emptyset = \text{derive_pode}(\mathbf{o}, \mathbf{s}, +\mathbf{a}))$
Nada é sobreposto	$\text{faça}(\mathbf{o}, \mathbf{s}, +\mathbf{a}) \leftarrow \text{derive_pode}(\mathbf{o}, \mathbf{s}, +\mathbf{a}) \&$ $(\emptyset = \text{derive_pode}(\mathbf{o}, \mathbf{s}, -\mathbf{a}))$ $\text{faça}(\mathbf{o}, \mathbf{s}, -\mathbf{a}) \leftarrow \text{derive_pode}(\mathbf{o}, \mathbf{s}, -\mathbf{a}) \&$ $(\emptyset = \text{derive_pode}(\mathbf{o}, \mathbf{s}, +\mathbf{a}))$

Quadro 2 - Exemplo de utilização da linguagem lógica para representar resoluções de conflitos.

2.2.3 Linguagem para Expressar Serviços de Segurança (LESS)

A linguagem LESS (MATTES et al., 2004) é uma iniciativa de estender as capacidades de *ASL* para os demais serviços de segurança, permitindo a representação lógica de regras e parâmetros de diferentes modelos de políticas. Em relação à *ASL*, as seguintes funcionalidades foram adicionadas:

- ✓ Para integridade e privacidade: possibilita representar modelos para a negociação dos mecanismos e parâmetros a serem utilizados em uma sessão de comunicação;
- ✓ Para autenticação: oferece meios para representar métodos e condições suportados para obtenção de elo de confiança no parceiro da comunicação;
- ✓ Para auditoria e irretratabilidade: possibilita representar a necessidade de realização de rotinas para a implantação desses serviços na realização de ações do sistema.

Além de adicionar as capacidades acima destacadas, LESS permite a representação de políticas em que os serviços de segurança possam interdependeer. Assim, uma decisão de controle de acesso pode ter como parâmetro o método de autenticação utilizado.

2.2.4 XACML

A Linguagem Extensível de Marcação para Controle de Acesso (*eXtensible Access Control Markup Language - XACML*) (XACML, 2006), padronizada pela entidade Padrões Abertos Avançados para a Sociedade da Informação (*Advancing Open Standards for the Information Society - OASIS*), tem como base a lógica formal e fornece um sistema genérico de autorização para atender às necessidades dos sistemas distribuídos atuais, permitindo que, por meio de um conjunto estruturado de regras e parâmetros, diferentes modelos de políticas possam ser representados e processados.

A Figura 1 traz o modelo de autorização do *XACML*, que conta com as seguintes entidades:

- Ponto de Efetivação de Política (PEP);
- Ponto de Decisão de Política (PDP);
- Ponto de Informação de Política (PIP);
- Ponto Administrativo de Política (PAP).

Figura 1 - Modelo de autorização XACML.

Posicionado entre a aplicação e os recursos, o PEP intercepta as requisições da aplicação e as envia para o PDP. O PDP realiza as decisões de acesso de acordo com as políticas de segurança escritas em PAP, utilizando os parâmetros relacionados ao sujeito, ao recurso e ao ambiente, obtidos junto ao PIP. Uma vez estabelecida, a decisão é enviada ao PEP para que possa bloquear ou permitir a ação.

As políticas de controle de acesso são estabelecidas e organizadas de forma hierárquica dentro de conjuntos de políticas e regras. Cada regra é uma condição de uso de um recurso-alvo e uma política possui um alvo, um conjunto de regras e, de forma opcional, obrigações.

A padronização XACML especifica a comunicação entre PDP e PIP. Contudo, não detalha a implementação do PIP, entidade que realiza uma tarefa importante: obter de forma segura os parâmetros relevantes para que o PDP realize as tomadas de decisões.

2.3 Efetivando Políticas de Segurança

Como já mencionado, os sistemas baseados em políticas separam a tomada de decisão da implementação. As políticas de segurança fornecem leis, regras e práticas que regulamentam como uma organização gerencia, protege e distribui informações, e os mecanismos oferecem meios para o seu cumprimento. Assim, além de compreender as políticas de segurança, os mecanismos devem possuir as funcionalidades necessárias para fazer cumprir as regras especificadas.

Em um sistema computacional, as aplicações atuam sobre um ambiente de execução, responsável por receber as requisições provenientes da aplicação, para interagir junto aos recursos envolvidos. Dessa forma, as políticas de segurança podem ser efetivadas pelo ambiente de execução, aplicação ou por uma combinação dos dois.

Sistemas nos quais as aplicações são responsáveis por efetivar parte ou todas as políticas de segurança oferecem, por um lado, maior flexibilidade, ao permitir que os desenvolvedores possam introduzir novos mecanismos de segurança e estabelecer novos pontos de execução para o cumprimento de políticas. Contudo, ao vincular a efetivação das políticas à aplicação, a adoção de novos modelos e a adição de funcionalidades de segurança podem ficar subordinadas a modificações nas aplicações. Outro inconveniente desse modelo é que exige a utilização de aplicações confiáveis, o que nem sempre é possível.

Na maior parte dos sistemas, os ambientes de execução, sistemas operacionais como Windows, Linux, Unix e Macintosh, ou plataformas como Java ou .net, são responsáveis pelo oferecimento dos serviços de segurança. Porém, utilizando mecanismos próprios, deixam a desejar em relação aos requisitos de flexibilidade e de interoperabilidade existentes nos modernos sistemas distribuídos como a grade computacional.

A evolução dos sistemas de segurança criou uma variedade de mecanismos e de modelos de políticas. Dessa maneira, a integração dessas tecnologias em um ambiente de execução flexível e compatível com as aplicações legadas é um desafio ainda não totalmente equacionado, tema abordado no presente trabalho.

2.4 Infra-Estruturas para Distribuição de Políticas

Esta Seção tem como objetivo apresentar as principais tecnologias para distribuição segura de políticas. A Seção 2.4.1 descreve o certificado de X.509, e as Seções 2.4.2, 2.4.3 e 2.4.4 apresentam, respectivamente, a Infra-estrutura de Gerenciamento de Privilégios (*Privilege Management Infrastructure - PMI*) (CHADWICK; OTENKO, 2002), a Infra-estrutura Simples de Chave Pública (*Simple Public Key Infrastructure - SPKI*) (WANG, 1998) e MyProxy. Por fim, a Seção 2.4.5 compara as soluções apresentadas.

2.4.1 Certificados X.509

Certificados X.509 são documentos assinados digitalmente e utilizados, normalmente, dentro de um contexto de uma Infra-estrutura de Chave Pública (ICP) (STALLINGS, 1998), para relacionar um sujeito com sua chave pública. Contudo, sua utilização pode ser estendida para conter outros atributos do sujeito.

A Figura 2 traz um exemplo de certificado digital no qual, além dos parâmetros padrões de um certificado como: número de série, nome da entidade ao qual o sujeito pertence, assegurador do certificado, chave pública do sujeito e assinatura digital, possui um campo atributo com as políticas de segurança relacionadas ao sujeito e codificadas por um determinado padrão.

```

Version: 1
Holder:
    BaseCertificateID: null
    EntityName: [CN=Leonardo Mattes, OU=LSI User, OU=Class 1, O=VT,
C=br]
    ObjectDigestInfo: null
Issuer:
    IssuerName: [CN=LSI Admin, OU=LSI, O=VT, C=br]
    BaseCertificateID: null
    ObjectDigestInfo: null
    Signature: SHA1withRSA, OID = 1.2.834.113522.1.1.2
    SerialNumber: SerialNumber: [ 45]
    AttrCertValidityPeriod:
    NotBeforeTime: jun 17 09:27:16 EDT 2006
    NotAfterTime: jun May 19 09:27:16 EDT 2007
    Attributes: 1

```

Figura 2 – Certificado digital.

2.4.2 PMI

A Infra-estrutura de Gerenciamento de Privilégio (*Privilege Management Infrastructure – PMI*) (CHADWIK; OTENKO, 2002) oferece especificações para distribuição de atributos de controle de acesso, cujos certificados X.509 são utilizados para armazenar direitos e permissões de entidades.

A estrutura funcional da *PMI* assemelha-se, em muitos pontos, a uma ICP. Se a ICP promove um forte elo entre as entidades e suas respectivas chaves públicas, a *PMI* oferece um forte elo entre o usuário e seus atributos de acesso. Para melhor entendermos a similaridade de funcionamento entre ICP e *PMI*, o Quadro 3 apresenta um relacionamento funcional das entidades presentes em ambas.

Conceito	Entidade ICP	Entidade <i>PMI</i>
Assegurador do certificado	Autoridade Certificadora (AC)	Autoridade Atribuidora (AA)
Recebedor de certificado	Sujeito Nome do Sujeito	Privilegiado nome do privilegiado
Conteúdo do certificado	Nome do sujeito e sua chave pública	Nome do privilegiado e um ou mais atributos para controle de acesso
Revogação	Lista de Certificados Revogados (LCR)	Lista de Certificados de Atributos Revogados (LCAR)
Raiz de confiança	AC Raiz	Fonte de atribuição (SOA- Source of Authority)
Autoridade subordinada	AC Subordinada	Autoridade atribuidora (AA)

Quadro 3 - Quadro comparativo entre as infra-estruturas ICP e PMI (CHADWICK; OTENKO, 2002).

A entidade primária de confiança em uma ICP é a base para o estabelecimento de confiança em seu domínio e é chamada de AC Raiz. Enquanto em uma *PMI* é chamada de Fonte da Autoridade (*Source of Authority - SOA*), na ICP, as ACs podem possuir ACs subordinadas de confiança. Similarmente, a fonte da autoridade também pode possuir AAs subordinadas. Na ICP, caso um usuário necessite revogar sua chave, a AC emite uma Lista dos Certificados Revogados (LCR). Numa *PMI*, se o usuário revoga suas permissões, a AA emite uma LCAR.

Resumindo, a *PMI* foi concebida a partir dos conceitos e entidades presentes na ICP com o intuito de oferecer suporte ao controle de acesso. Na prática, as duas infra-estruturas possuem estruturas similares e oferecem serviços distintos:

- A ICP é uma infra-estrutura para utilização dos serviços de criptografia, estabelecendo um elo seguro entre entidades e suas respectivas chaves públicas;
- A *PMI* oferece suporte para controle de acesso, permitindo que os atributos de acesso de entidades sejam distribuídos de forma segura.

2.4.3 SPKI

A Infra-estrutura Simples de Chave Pública (*Simple Public Key Infrastructure - SPKI*) (WANG, 1998) propõe uma simplificação da ICP para sua utilização em sistemas de controle de acesso. Ao invés de utilizar nomes e a estrutura hierárquica de AC, a *SPKI* baseia-se exclusivamente em chaves públicas e assinaturas digitais, permitindo que os usuários assinem seus certificados e deleguem seus privilégios de acesso por um determinado período.

A estrutura simplificada do certificado *SPKI* mostrado na Figura 3 permite sua utilização para armazenar atributos dos diferentes modelos de controle de acesso e também de chaves públicas de usuários. Os campos fixos de delegação e de datas indicam, respectivamente, a possibilidade do sujeito de delegar os poderes contidos na credencial e o período de validade do certificado.

Figura 3 - Modelo de certificado *SPKI* (WANG, 1998).

A *SPKI* tem como principais atributos a flexibilidade e a simplicidade que permitem sua fácil implementação, podendo ainda contar com o auxílio de uma ICP, responsável por fornecer um elo mais forte entre entidades e chaves públicas.

2.4.4 MyProxy

O serviço de gerenciamento de credenciais MyProxy (Novotny; Tuecke, 2001) oferece um repositório de certificados utilizados para autenticar usuários em grades computacionais. No processo de gerenciamento de credenciais do MyProxy, na fase de iniciação, o usuário fornece sua credencial de acesso junto a uma frase secreta,

para que, posteriormente, a credencial fique disponível ao usuário ao revelar a frase secreta.

Além de armazenar as credenciais de acesso, o MyProxy pode ser utilizado para efetuar os processos de autenticação pelo usuário. Esse processo está demonstrado na Figura 4 em que, ao se conectar ao sítio, o usuário fornece os dados para delegar o processo de autenticação para o servidor MyProxy. Como os dados de autenticação são criptografados pela frase secreta, o MyProxy reconhece o pedido como proveniente do usuário e utiliza a credencial de acesso e a chave privada para prosseguir com a autenticação.

Figura 4 - Processo de delegação de autenticação utilizando MyProxy.

2.4.5 Comparando as Tecnologias para Distribuição de Políticas

As infra-estruturas de distribuição de políticas estudadas são utilizadas em sistemas distribuídos para promover de forma segura a relação entre sujeitos e atributos de segurança, porém foram desenvolvidas para diferentes contextos e possuem características distintas.

De forma geral, entre as plataformas estudadas a *PMI* é a mais complexa e rígida, contando com uma estrutura hierárquica de confiança, baseada em nomes, certificados digitais e chaves públicas.

Quando comparada com a *PMI*, a *SPKI* é mais simples e flexível, sendo as relações de confiança estabelecidas de forma manual e os parâmetros de controle de acesso vinculados às chaves públicas e não às entidades.

O serviço *MyProxy* foi concebido especificamente para ambientes de grades

computacionais e oferece um repositório para efetivar os processos de autenticação e validação de credenciais de acesso em nome do usuário.

2.5 Conclusões

Sem dúvida, a segurança em redes computacionais é uma área de enorme importância, contando com inúmeros mecanismos e tecnologias. Diante da complexidade do tema, este capítulo descreveu apenas os conceitos e mecanismos que estão diretamente relacionados ao presente trabalho.

As lógicas formais e linguagens de políticas de segurança descritas neste capítulo têm como objetivo oferecer um primeiro passo na integração de sistemas heterogêneos, que é a representação dos diferentes modelos e parâmetros das políticas envolvidas.

Foram apresentadas infra-estruturas para distribuição segura de políticas com características distintas que possuem seu uso recomendado para diferentes contextos ou finalidades.

Resumindo: os conceitos e as tecnologias de segurança em ambientes distribuídos até aqui apresentados estão direta ou indiretamente presentes no modelo de segurança proposto por esta tese.

3 Segurança em Grades Computacionais

Este capítulo traz um panorama das tecnologias de grade computacional com ênfase nas vulnerabilidades e requisitos de segurança. O capítulo apresenta também padrões e requisitos para o estabelecimento de plataformas de grade computacionais interoperáveis.

Em seguida, diferentes tecnologias de segurança para grades computacionais são analisadas e comparadas.

3.1 Grade Computacional

Com a popularização da Internet, os últimos 15 anos foram marcados pelo rápido desenvolvimento das tecnologias da Ciência da Informação e pela crescente demanda por recursos computacionais. Neste cenário, a computação distribuída vem cumprindo um papel importante, tanto por aumentar a capacidade de processamento, distribuindo tarefas grandes para serem executadas paralelamente em diversas instâncias, como também por compartilhar e integrar recursos. Neste contexto, surge o termo “grade computacional” como uma proposta de evolução das soluções existentes.

O termo “grade computacional” faz referência a uma classe de sistemas cuja principal característica é o compartilhamento de recursos e serviços de forma integrada e dinâmica entre múltiplos domínios. Seu objetivo é proporcionar um melhor aproveitamento e gerenciamento dos recursos disponíveis e, conseqüentemente, a redução nos custos associados (FOSTER, 2005).

Um típico cenário de grade computacional envolve VOs, que são associações temporárias entre entidades para que possam compartilhar recursos, habilidades e competências. Uma vez estabelecida, a VO faz uso da grade computacional para oferecer as entidades associadas aos seguintes serviços:

1. Serviço computacional: proporciona a execução segura e remota de tarefas distribuídas;
2. Serviço de dados: provê a distribuição e o gerenciamento de dados que podem utilizar redundância para incrementar desempenho e

disponibilidade;

3. Serviço de aplicação: oferece o gerenciamento e o acesso remoto a aplicações e bibliotecas;
4. Serviço de informação: permite a pesquisa, a extração e a apresentação de dados, utilizando, nesse sentido, os serviços de dados e/ou aplicação;
5. Serviço de conhecimento: fornece aos usuários meios para adquirir, usar e publicar conhecimentos visando um objetivo.

Um ambiente de grade computacional completo exige uma plataforma de serviços integrados que incluam segurança, informação, diretório, alocação de recursos, mecanismo de pagamento, gerenciamento de execução e agendamento. A Figura 5 apresenta a arquitetura de grades computacionais do projeto Globus, composta pelas seguintes camadas:

- **Tecido:** camada de interface de alto nível, operando entre os serviços da grade e os recursos locais, tais como sistema de arquivos e interfaces de redes;
- **Conectividade:** fornece os protocolos para comunicação e autenticação dos processos da grade computacional;
- **Recursos:** construídos em cima da camada da conectividade, define os protocolos e serviços de negociação, inicialização, monitoramento, controle e pagamento dos processos individuais ou agrupados;
- **Coletivo:** diferentemente da camada de recursos, que é focada em somente um sítio, a camada de coletivo é responsável por fazer interagir e gerenciar múltiplos sítios, a fim de proporcionar um melhor aproveitamento da infra-estrutura distribuída disponível;
- **Aplicação:** compreende a camada final do usuário que tem à sua disposição uma interface de alto nível para gerenciar os recursos da grade, e pode ser construída utilizando diferentes linguagens de programação.

Figura 5 - Modelo de camadas da plataforma Globus Toolkit (FOSTER, 2005).

A tecnologia de grade computacional, embora ainda em processo de maturação, já conta com diversos casos de usos, aplicações e plataformas. Como a interoperabilidade é um dos objetivos primários da grade e diante da diversidade apresentada, a padronização de protocolos e serviços tornou-se uma tarefa crítica no desenvolvimento de suas tecnologias. No intuito de oferecer resposta a esses requisitos, a iniciativa Fórum Aberto de Grade (*Open Grid Forum - OGF*) (OGF, 2007) traz padrões para troca de mensagens, arquitetura, serviços e protocolos relacionados a tecnologias de grade.

Entre as especificações oferecidas pela *OGF*, destaca-se a utilização de serviços web para comunicação. O Protocolo Simples de Acesso a Objetos (*Simple Object Access Protocol - SOAP*), padrão de serviços web fornecido pela entidade W3C (W3C, 2007), especifica esquemas *XML* que são utilizados para descrever e utilizar serviços de rede, facilitando assim, a integração entre sistemas. Desta maneira, o serviço web é crescentemente utilizado como tecnologia de comunicação em grades computacionais para interconectar domínios e aplicações.

3.2 Requisitos de Segurança para Grades Computacionais

Ao oferecer maior integração entre domínios, as grades computacionais aumentam também a exposição a riscos. Atualmente, as tecnologias de grade estão

sendo utilizadas quase que exclusivamente em ambientes acadêmicos. Para ser adotada em larga escala em sistemas corporativos, faz-se necessário o desenvolvimento de infra-estruturas seguras, que ofereçam respostas aos requisitos específicos da grade, sem comprometer as estruturas e os serviços existentes.

A segurança computacional é um tema complexo que vem sendo estudado e desenvolvido já há algum tempo e conta com inúmeros mecanismos. Desta maneira, no intuito de minimizar os custos associados, o provimento de segurança em grades computacionais deve fazer uso dos conceitos e tecnologias existentes ao máximo.

Diferentemente de um sistema corporativo tradicional centralizado, a grade computacional é gerenciada em VO, que envolvem diferentes domínios lógicos e tecnológicos estabelecidos de forma dinâmica. A Figura 6 traz uma representação da estrutura de uma VO de domínio sobre domínios. Deste modo, um dos requisitos da grade vem de sua estrutura, que exige que as relações de confiança entre domínios e usuários sejam estabelecidas e gerenciadas de forma dinâmica.

Uma das principais características da grade computacional é a heterogeneidade tecnológica. Como, nessas condições, um modelo de segurança homogêneo, que se sobreponha às soluções existentes, é inviável devido aos custos de sua implementação, cabe o fornecimento de uma infra-estrutura flexível, que faça uso dos diferentes mecanismos de segurança existentes.

O armazenamento, o gerenciamento e a distribuição de políticas devem ser orientados conforme as características da grade computacional de descentralização, permitindo que (DAMIANI; VIRMECATI; SAMARATI, 2005) (COETZEE; ELOFF, 2003):

1. Um usuário, ao inserir um novo serviço, possa também definir as políticas de uso do mesmo;
2. A VO estabeleça diferentes níveis de acesso para seus associados;
3. Um sítio, membro de uma VO, possa estabelecer suas próprias regras para controlar os recursos que disponibiliza.

Figura 6 - Estrutura dinâmica de relações de confiança das Organizações Virtuais envolvendo múltiplos domínios (WELCH et al., 2003).

Dentro dos serviços oferecidos pela grade, a instanciação remota e dinâmica de tarefas e serviços necessita ser analisada de forma mais específica, do ponto de vista da segurança, uma vez que tal funcionalidade, se explorada de forma maliciosa ou utilizada com aplicações contendo falhas de segurança, pode permitir que ataques sejam realizados a partir de uma posição privilegiada, dentro da rede corporativa.

Como exemplo, podemos imaginar as empresas “A” e “B” que criam uma VO para poder compartilhar recursos e serviços em um determinado projeto colaborativo. A criação dinâmica de um serviço por “A” em “B” permite a utilização dos servidores da empresa “B”, que estão momentaneamente ociosos, e/ou melhor, explorar a topologia de rede dada, diminuindo o tráfego entre os domínios. Contudo, o programa instalado em “B” pode ter sido desenvolvido de forma maliciosa por um funcionário de “A”, possibilitando-lhe usufruir de uma posição privilegiada de seu programa, dentro da rede corporativa de B, para descobrir e explorar possíveis vulnerabilidades de segurança.

A solução do problema acima descrito não pode ser equacionada de forma simples, por meio de uma solução única. Porém, uma forma de minimizar os

possíveis danos consiste em aplicar o princípio do privilégio mínimo, que atribui a uma tarefa somente os privilégios necessários para o seu cumprimento, e não mais que isso.

3.3 Modelo de segurança OGSA

O oferecimento de segurança em ambientes de grade computacional envolve a integração de recursos heterogêneos e dinâmicos. Como na Ciência da Computação a padronização é o meio de perseguir a interoperabilidade, o oferecimento de padrões é uma tarefa crítica para o desenvolvimento de grades computacionais abertas, que operem em conjunto.

O OGF oferece a Arquitetura de Serviços de Grades Abertas (*Open Grid Services Architecture - OGSA*) (NAGARATNAM et al., 2003) como padrão a ser seguido no desenvolvimento de plataformas para grades computacionais. Em relação à segurança, a arquitetura OGSA define serviços web padronizados para facilitar a administração, expressão, publicação, descobrimento, comunicação, verificação e cumprimento de políticas, prevendo os seguintes serviços padronizados:

- Processamento de credenciais: processa e valida chaves digitais (*tokens*) de autenticação;
- Autorização: requerido para permitir ou negar o acesso a um recurso da grade. As tomadas de decisões são realizadas analisando as políticas de segurança, a identidade do requisitante e os atributos associados aos recursos;
- Conversão de credenciais: permite a interoperabilidade entre diferentes domínios, por meio da conversão de credenciais;
- Auditoria: armazena e distribui informações relevantes sobre eventos.

A Figura 7 ilustra um exemplo prático de uso dos serviços padronizados do OGSA, em que um cliente de um domínio requisita um recurso de outro domínio, por meio dos seguintes passos:

1. O usuário autentica-se junto ao servidor de autenticação;
2. Em seguida, envia ao intermediário uma requisição para utilizar serviços da grade;

3. O intermediário recebe a chave eletrônica (*token*) de autenticação e adquire os parâmetros do usuário junto ao servidor de atributos para confeccionar o certificado;
4. Caso não haja relações de confiança direta entre os dois domínios, uma entidade intermediária é requisitada para mapear o usuário, gerando outro certificado que seja válido no domínio do recurso;
5. O provedor do recurso requisitado valida o usuário e seus atributos para que possa utilizar os serviços oferecidos;
6. O ambiente de execução faz uso dos serviços de autorização e de outros mecanismos locais para cumprir as políticas de segurança.

Figura 7 - Modelo de segurança OGSA (NAGARATNAM et al., 2002).

O ambiente de execução é o primeiro e muitas vezes o único obstáculo entre a aplicação e os recursos. O padrão da OGSA, apesar de não especificar em detalhes, recomenda a utilização de ambientes de execução capazes de fazer cumprir as políticas de segurança e integrar os diferentes mecanismos já existentes no sítio de uma VO. Sendo assim, o ambiente de execução é a entidade responsável pelo cumprimento de boa parte das funcionalidades de segurança.

3.4 Plataformas de Segurança em Ambientes de Grade Computacional

Esta Seção contém uma visão geral da segurança em diferentes plataformas

de grade computacional, comparando-as principalmente aos temas diretamente relacionados a esta tese.

3.4.1 Globus Toolkit 4

O *Globus Toolkit 4 (GT4)* consiste em um conjunto de ferramentas, cada qual implementando um determinado serviço que, quando trabalhando de forma integrada, oferece uma infra-estrutura para grades computacionais.

O Serviço de Autorização Comunitária (*Community Authorization Service - CAS*) (PEARLMAN et al., 2003) do *GT4* faz uso de um repositório centralizado em domínios e de certificados digitais X.509, para o estabelecimento dinâmico de relações de confiança e para a distribuição e delegação segura de atributos de segurança. Segue um modelo no qual, antes de usufruir os serviços da *VO*, o usuário realiza um processo de autenticação junto à *CAS* e recebe uma credencial de acesso, contendo sua chave pública, entre outros atributos de segurança, conforme exemplificado pela Figura 9. A credencial de acesso é utilizada em todas as requisições de recursos para que em um serviço *single sign-on* os servidores possam autenticar corretamente e obter os atributos de usuário, sem sua intervenção direta. A curta validade da credencial de acesso obriga o usuário a renovar periodicamente seu certificado com seus atributos de segurança, que podem ter sido alterados.

Figura 8 - Exemplo de credencial de acesso contendo a chave pública e atributos de segurança do usuário (PEARLMAN et al., 2003).

Os serviços do *GT4 GridFTP* e de Localização de Réplicas (*Replica Location Service - RLF*) trabalham em conjunto para armazenar e distribuir dados entre múltiplas fontes com redundância, agendamento e alta performance. Tais serviços contam ainda com diferentes algoritmos de criptografia para privacidade e podem trabalhar de forma integrada ao *CAS*, com o propósito de realizar o controle de acesso.

O Serviço Web para Alocação e Gerenciamento de Recursos de Grade (*Web Service Grid Resource Allocation and Management - WS GRAM*) permite a instanciação remota de tarefas, processo em que, a partir de uma requisição para a criação de uma tarefa, o *WS GRAM* prossegue por meio dos seguintes passos:

- Promove possíveis transferências de arquivos úteis à tarefa a ser realizada;
- Por meio do uso de rotinas, inicia o processo utilizando uma conta local;
- Oferece uma interface para que o usuário acompanhe o estado do processo;
- Ao final, caso haja necessidade, realiza novas transferências de arquivos.

No processo de requisição de instanciação de tarefas, o usuário fornece um arquivo que segue a sintaxe da Linguagem para Especificação de Recursos (*Resource Specification Language - RSL*), que contém uma série de informações como: o nome do executável a ser utilizado, bibliotecas necessárias, parâmetros de

inicialização, arquivos que precisam ser transferidos antes do início da tarefa etc.

No processo de inicialização, o serviço *WS GRAM* faz uso de um arquivo de configuração local para mapear usuários da *VO* em contas do sistema local. Dessa forma, a tarefa submetida passa a usufruir todos os privilégios da conta local utilizada, sendo o controle de acesso realizado pelo sistema operacional.

3.4.2 SGE

A plataforma Motor de Grade Computacional Sun (*Sun Grid Engine - SGE*) (SGE, 2007) tem como objetivo integrar ambientes em corporações, não se preocupando em integrar diferentes domínios. Com esse intuito, oferece uma solução flexível para o gerenciamento de agrupamentos (*cluster*), junto a uma série de funcionalidades das grades computacionais.

O *SGE* oferece um serviço de execução remota de tarefas, no qual o escalonamento da execução nos sítios a serem utilizados é estabelecido por políticas que ponderam sobre os perfis das tarefas e o estado da grade. Assim, toda tarefa é associada a um perfil de execução, conforme seu nível de criticidade e as características do usuário que a submete.

Entre os objetivos principais do *SGE* não se encontra a segurança, pois espera-se que a grade atue dentro de um sistema corporativo e confiável. Contudo, existem meios para integrar alguns serviços de segurança, tais como: a) autenticação – Kerberos (KERBEROS, 2007); b) privacidade e integridade de dados em trânsito – Camada de Comunicação Segura (*Secure Socket Layer - SSL*); c) auditoria – mecanismo próprio.

3.4.3 Legion

O Legion (BRIAN et al., 2001) oferece uma plataforma de grade computacional orientada a objetos em que os diferentes recursos, entidades e mecanismos são encapsulados e acessíveis por meio de métodos de alto nível, ocultando assim detalhes de implementação, como protocolos e interações. A criação de novas classes e objetos é realizada em cima de um ambiente de execução próprio, a Biblioteca de Execução Legion (*Legion Runtime Library - LRTL*),

que oferece ao usuário uma série de classes para a realização de funcionalidades como: armazenamento e acesso remoto de dados, agendamento, instanciação e gerenciamento remoto de tarefas.

Os serviços de autenticação e distribuição segura de atributos são garantidos por meio de certificados X.509, nos quais relações de confiança interdomínios podem ser estabelecidas e configuradas levando em consideração os atributos dos certificados e a Autoridade Certificadora (CA). Os outros serviços de segurança, como controle de acesso, auditoria, integridade, privacidade e não repudição, não são oferecidos, ficando a cargo das aplicações realizarem suas próprias funcionalidades de segurança.

3.4.4 *OurGrid*

A plataforma de grade computacional *OurGrid* (OurGrid, 2007) segue um modelo de relações de confiança ponto a ponto, para que sítios possam submeter e controlar tarefas em outros sítios de sua confiança. Entre as características do *OurGrid*, podemos destacar a leveza de sua infra-estrutura enxuta e a utilização de políticas para escalonamento de tarefas que permitem oferecer melhor qualidade de serviço para nós mais bem pontuados.

O *OurGrid* possui um mecanismo de segurança muito simples, cujas relações de confiança são estabelecidas e administradas ponto a ponto, e em que a autenticação é realizada pelo mecanismo *secure shell* (SSH), que faz uso de criptografia de chave pública para uma autenticação baseada em identificador (*login*) e senha. O controle de acesso é realizado pelo sistema operacional, conforme as características da conta local utilizada pelo serviço *OurGrid*.

3.4.5 *InteGrade*

InteGrade (GOLDCHLEGER et al., 2004) é uma plataforma destinada à utilização de recursos computacionais não dedicados, fazendo uso da capacidade ociosa dos sítios associados.

O *InteGrade* possui um arquitetura orientada a objetos, utiliza tecnologias CORBA como infra-estrutura de comunicação e oferece suporte a aplicações de

paralelismo não trivial. Para gerenciar a execução de tarefas e impedir a sobrecarga, possui um mecanismo de escalonamento que monitora o estado dos nós utilizados.

A Figura 9 representa a arquitetura do *InteGrade*, que contempla os seguintes componentes:

- Gerenciador do Aglomerado, responsável pela administração da execução das aplicações;
- Nós Dedicados e Compartilhados, disponibilizados para executar as tarefas paralelas;
- Nó do Usuário, que contém mecanismos para submeter e controlar aplicações.

Figura 9 – Arquitetura InteGrade.

Um repositório armazena diferentes versões dos binários das aplicações para que, posteriormente, sejam executados conforme a disponibilidade da grade. PINHEIRO JÚNIOR, VIDAL E KON (2005) propõem o uso da interface de programação Genérica de Serviços de Segurança (*Generic Security Services - GSS*) (KABAT; UPADHYAY, 2000) para acrescentar segurança ao gerenciamento das aplicações da InteGrade, adicionando confiabilidade e autenticação aos processos de comunicação, além de garantir a autenticidade das aplicações por meio de um esquema de assinatura digital.

No intuito de estabelecer um modelo de confiança descentralizado mais flexível no InteGrade, PINHEIRO JÚNIOR E KON (2006) propõem uma extensão das Infra-estruturas Simples de Criptografia de Chave Pública / de Segurança Distribuída (*Simple Public Key Infrastructure / Simple Distributed Security Infrastructure - SPKI/SDSI*) (RONALD; BUTLER, 1996). No modelo proposto, as relações de confiança são baseadas em opiniões, tuplas contendo três índices que indicam a crença, descrença e incerteza sobre um sujeito. Para estabelecer e alterar a opinião sobre um sujeito, além de analisar seu comportamento, o modelo considera opiniões emitidas por outras entidades. Futuras políticas de controle de acesso podem realizar regras que tenham como base os índices de opinião para permitir ou negar ações de sujeitos.

3.5 Analisando as Soluções de Segurança para Grades Computacionais

Para melhor compreender o estado das tecnologias de grade computacional, principalmente em relação à segurança, esta Seção apresenta uma análise comparativa entre as plataformas acima descritas. O quadro 4 sumariza as principais diferenças entre as plataformas OurGrid, *Globus Toolkit 4*, *Sun Grid Engine*, Legion e InteGrade, em que, com base nos requisitos de segurança da Seção 2.1 e nas recomendações da OGSA, foram utilizados os seguintes parâmetros de comparação:

- *Relações de confiança*: o GT4 se destaca por oferecer resposta ao requisito de grade referente ao estabelecimento de relações de confiança entre múltiplos domínios, de forma flexível. Na infra-estrutura ICP, utilizada pelo SGE e pelo Legion, as relações de confiança são estabelecidas levando em consideração a Autoridade Certificadora (AC) e os atributos dos certificados, o que pode não condizer com a estrutura de uma VO mais flexível e dinâmica. Na plataforma OurGrid, cada sítio estabelece localmente em quais entidades deve confiar, mecanismo simples que torna difícil o gerenciamento de VOs com grande número de associados. A proposta para relações de confiança do InteGrade é inovadora, por apresentar um modelo baseado em opiniões, como já abordado;

- *Interoperabilidade com mecanismos de segurança:* o GT4 utiliza um serviço de autenticação e delegação de privilégios interoperável com diferentes mecanismos de autenticação, além da funcionalidade de mapeamento de usuários da VO em contas locais, permitindo que o controle de acesso seja realizado localmente - funcionalidade também presente no OurGrid. As plataformas estudadas não prevêm nenhuma outra funcionalidade ou interface que permita interoperar com mecanismos de segurança já existentes, como IDS, Firewall etc;
- *Gerenciamento dinâmico e descentralizado de políticas de segurança:* o projeto GT4 oferece o serviço CAS para o armazenamento e a distribuição de atributos de controle de acesso válidos em uma VO. Nos projetos OurGrid e InteGrade, o controle de acessos é gerenciado e realizado totalmente no sítio da VO. Segundo o modelo de SGE e Legion, as políticas de segurança são estabelecidas e gerenciadas no sítio, que pode levar em consideração algum atributo do certificado do usuário.
- *Ambiente de execução:* no GT4 as aplicações web são hospedadas no contêiner do Apache Tomcat (APACHE TOMCAT, 2007) e as tarefas diretamente no sistema operacional. O InteGrade, o SGE e o OurGrid utilizam somente o sistema operacional como ambiente de execução; o Legion oferece um ambiente de execução próprio e homogêneo sobre diferentes sistemas operacionais, permitindo abstrair as peculiaridades ambientais, embora não ofereça nenhuma funcionalidade de segurança.
- *Uso do padrão OGSA:* entre as plataformas estudadas o GT4 é a única que utiliza os protocolos propostos pela OGSA e se esforça para seguir suas recomendações.

Como pode ser observado no Quadro 4, não existe um conceito único relacionado à grade computacional. As diferentes soluções estudadas e comparadas nesse capítulo não diferem somente na implementação, mas também sob a ótica de seus objetivos e modelos de segurança adotados.

Parâmetros	OurGrid	GT4	SGE	InteGrade	LEGION
Relações de confiança	Ponto a ponto	Interdomínios	Kerberos.	Cadeias de opiniões	ICP
Interoperabilidade com mecanismos de segurança	Mapeamento em conta local	Mapeamento em conta local	Não prevê.	Não prevê	Não prevê
Gerenciamento de políticas de segurança	Somente políticas locais	Combina atributos distribuídos com políticas locais	Políticas locais mais certificados x.509	Não prevê	Políticas locais mais certificados x.509
Ambiente de execução	Sistema operacional	Sistema operacional para tarefas e Tomcat Contêiner para serviços	Sistema operacional	Sistema operacional	Sistema próprio
Padrão OGSA	Não Implementa	Implementa	Não Implementa	Não Implementa	Não Implementa

Quadro 4 - Comparação entre as plataformas de grade estudadas.

O padrão *OGSA*, apesar de ser uma tentativa de integrar os diferentes requisitos de grade computacional, é implementado e seguido somente pelo *GT4*. Contudo, a contribuição do *OGSA* não pode ser minimizada. Para a realização de um ambiente de grade integrado e heterogêneo a padronização é fundamental. Dessa forma, espera-se que com o amadurecimento das tecnologias de grade computacional aumente a importância da padronização e a especificação *OGSA* seja difundida.

Entre as soluções estudadas, o *GT4* é a plataforma que melhor atende às necessidades discutidas neste capítulo, no que diz respeito à segurança de grades computacionais abertas e heterogêneas, sendo possível destacar as seguintes funcionalidades do *GT4*: possui um modelo flexível de estabelecimento de relações de confiança interdomínios que melhor se adapta às características da *VO*; obedece à especificação *OGSA*; oferece um modelo flexível de políticas que combina o uso de políticas da *VOs* e locais. Apesar de não preencher todos os requisitos em relação à interoperabilidade, o *GT4* já realizou passos importantes nesse sentido, como a utilização de diferentes protocolos de autenticação e o oferecimento de

serviços para armazenamento e distribuição de diferentes modelos de políticas de segurança.

Quanto aos requisitos de segurança vistos na Seção 3.2 e às recomendações do *OSSF*, podem-se destacar dois requisitos que ainda não foram sanados:

Primeiro: a interoperabilidade entre diferentes mecanismos de segurança existentes. As soluções apresentadas neste capítulo contemplam o uso de diferentes protocolos de autenticação; o controle de acesso integrado ao sistema operacional, e a infra-estrutura para armazenamento e distribuição de atributos de segurança. Contudo, falta estudar uma operabilidade mais ampla, que permita o emprego dos diferentes modelos de controle de acesso e mecanismos de segurança existentes, tais como *Firewall* e *IDS*.

Segundo: outro tema de grande importância e diretamente relacionado ao primeiro item é a oferta de um ambiente de execução complexo e flexível, que gerencie e utilize diferentes mecanismos.

3.6 Soluções de Controle de Acesso em Grades Computacionais – Trabalhos Relacionados

Esta Seção aprofunda a discussão da segurança em grades computacionais, focando nas soluções e propostas de serviços de controle de acesso que constituem o objeto de estudo da presente pesquisa. Os trabalhos relacionados são apresentados em duas categorias, de acordo com seus objetivos: uma, voltada para as infra-estruturas distribuídas e flexíveis de controle de acesso em serviços de rede, e outra, para a instanciação remota de tarefas.

3.6.1 Infra-estruturas de Controle de Acesso em Serviços de Rede

Uma parte importante das capacidades das grades computacionais é acessível por meio de serviços de rede, envolvendo recursos e usuários de múltiplos domínios. Esta Seção contém as principais propostas relacionadas a sistemas de controle de acesso para serviços de rede de grades computacionais.

Serviço de Autorização Comunitária (*CAS - Community Authorization Service*)

O CAS (PEARLMAN et al., 2003) foi desenvolvido pelo projeto Globus para prover um sistema de autorização flexível para grades computacionais de larga escala.

O serviço CAS faz uso dos mecanismos de segurança do GT4 para prover um sistema *single sign-on* baseado em autenticação de chave pública, delegação de privilégios e mapeamento de usuários. No modelo adotado, a sobrecarga administrativa é dividida, de modo que o dono do serviço a ser protegido define as regras de acesso e o administrador da VO, por meio da infra-estrutura do GT4, gerencia as relações de confiança e os atributos dos usuários.

A Figura 10 traz o processo de autorização da CAS em serviços de uma grade computacional que ocorre por meio dos seguintes passos:

1. Com o seu certificado X.509, o usuário da VO autentica-se no servidor CAS;
2. Baseado nas políticas estabelecidas pelo administrador da VO, o CAS cria uma credencial de acesso contendo os privilégios do usuário;

O usuário utiliza a credencial obtida para requisitar serviços na VO. Com base nas regras de acesso definidas pelos donos dos recursos e os privilégios contidos na credencial de acesso, as requisições do usuário são negadas ou permitidas.

Figura 10 – Uso da CAS no processo de autorização (PEARLMAN et al., 2003).

Para a representação dos privilégios de acesso, a CAS conta com formato próprio não padronizado, característica que pode gerar problemas de interoperabilidade com outras tecnologias de grade computacional. Atualmente, somente o serviço *GridFTP* do *GT4* oferece suporte ao uso da CAS.

Sistema de Autorização Akenti

O sistema Akenti (OTENKO E CHADWICK, 2007); (THOMPSON et al., 2003) oferece mecanismos de administração e efetivação de políticas para controle do uso de recursos distribuídos em grades computacionais. Esse sistema faz uso de uma linguagem própria, baseada em *XML*, para representação de regras e atributos de segurança distribuídos em certificados digitais.

A Figura 11 representa o modelo de autorização Akenti, no qual, após autenticado, o cliente envia ao intermediário a solicitação de um serviço, que é encaminhada ao servidor de autorização. Na seqüência, este obtém os certificados de políticas para a tomada de decisão. Caso autorizado, o recurso é disponibilizado

ao usuário. Para a tomada de decisão, o servidor de autorização Akenti obtém três tipos de certificados, contendo: regras de acesso do recurso; parâmetros de segurança do usuário; condições de uso que devem ser obedecidas.

Figura 11 – Seqüência do processo de autorização Akenti (Thompson et al. 2003).

Os certificados digitais utilizados no processo de autorização podem ser armazenados em um repositório central ou oferecidos pelos usuários. O Akenti oferece ferramentas e serviços administrativos para criar, alterar e armazenar certificados digitais com as políticas de controle de acesso.

Plataforma de Autorização com Múltiplas Políticas

LANG et al. (2006) propõem o uso de interfaces de comunicações para integrar sistemas de controle de acesso, permitindo que cada sítio da VO especifique suas fontes de autorização para proteger seus recursos. O uso desta plataforma possibilita mesclar políticas da VO com as políticas locais.

No processo de autorização, antes de processar uma requisição de um serviço, uma consulta é encaminhada ao Ponto de Decisão de Políticas (PDP) principal. Este PDP coleta decisões nas diferentes fontes de autorização (PDPs

secundários).

Por meio da edição de um arquivo de configuração, o administrador do sítio define quais os PDPs secundários que devem compor o processo de autorização. A comunicação entre o PDP principal e os secundários ocorre por meio de uma interface de programação, o que permite adicionar ao processo diferentes tecnologias de controle de acesso e fontes.

A Figura 12 traz, a título de exemplo, um PDP principal que conta com três PDPs secundários, integrando as tecnologias de controle de acesso CAS, Akenti e outra específica do domínio do sítio.

Figura 12 – Exemplo de estrutura com múltiplas políticas.

A utilização de arquivos de configurações locais para definir os PDPs a serem utilizados em um sítio não oferece a flexibilidade administrativa necessária em uma VO, onde usuários, recursos e domínios podem ser associados de forma dinâmica e, conseqüentemente, mudar também as fontes de autorização desejáveis.

Outra limitação da plataforma é a não definição de como as fontes de autorização, ou PDPs, podem obter as políticas e parâmetros necessários à tomada de decisões.

3.6.2 Controle de Acesso em Instanciação Remota de Tarefas

Os trabalhos relacionados ao controle de acesso de tarefas instanciadas remotamente têm como objetivo estabelecer ambientes de execução seguros, atuando justamente na área onde as plataformas acima não são aplicáveis.

Gerenciamento Dinâmico de Contas do Sistema Operacional

Como já abordado, o serviço de instanciação remota de tarefas do *GT4*, o *WS GRAM*, faz uso de um arquivo local para mapear usuários da *VO* em contas do sistema operacional, atribuindo aos mecanismos locais a tarefa de exercer o controle de acesso. Tal solução gera uma sobrecarga administrativa considerável em *VOs* com grande número de sítios e usuários. Os trabalhos a seguir apresentados propõem mecanismos para o gerenciamento dinâmico de contas do sistema operacional, no intuito de atribuir maior flexibilidade e diminuir a sobrecarga administrativa no serviço *WS GRAM*.

KEAHEY, RIPEANU e DOERING (2004) propõem um mecanismo rudimentar em que, antes da inicialização de uma tarefa, o arquivo de configuração é alterado de forma a dirigir o mapeamento na conta local. A alteração do arquivo de configuração leva em consideração os atributos de segurança da credencial do usuário que submeteu a tarefa.

O projeto PRIMA (LORCH; KAFURA, 2006) adota um modelo baseado em certificados X.509, contendo privilégios de acesso descritos segundo o padrão *XACML*. Os certificados são gerados por um serviço próprio oferecido e permite-se que os usuários deleguem seus privilégios para outras entidades. Na instanciação de tarefas são criadas e configuradas contas no sistema operacional, de acordo com os privilégios contidos nos certificados do usuário. Para a efetivação das regras de acesso são utilizados os mecanismos do sistema operacional, que permitem restringir o acesso que cada conta deve possuir em relação aos arquivos, diretórios do sistema, números *IP* e portas *TCP* da rede.

Mecanismos Baseados em Máquinas Virtuais

Mecanismos do tipo máquinas virtuais implementam diferentes ambientes de execução dentro de uma mesma máquina. Do ponto de vista da segurança, a utilização de máquinas virtuais permite definir que recursos do sistema operacional serão acessíveis no ambiente de execução fornecido, isolando a aplicação e diminuindo os possíveis danos que ela possa causar. Esta Seção analisa soluções de segurança para instanciação remota de tarefas em grades computacionais, baseadas em máquinas virtuais.

KEAHEY et al. (2004) propõem um serviço para criação de máquinas virtuais utilizando diferentes tecnologias. Na proposta, a cada submissão de tarefa são analisados os parâmetros contidos na credencial de acesso do usuário para a realização de uma máquina virtual configurada, com acesso parcial aos recursos do sistema.

DODONOV, QUAINI e QUARDIÃ (2004) propõem a configuração de máquinas virtuais Linux para hospedagem de tarefas remotas. Neste modelo, o administrador dos sítios configura a máquina virtual a ser utilizada, atribuindo os arquivos locais, os endereços e as portas *TCP/IP* que podem ser acessíveis. A falta de flexibilidade desta solução é notada uma vez que, independentemente das características do usuário ou da tarefa, as aplicações instanciadas remotamente em um sítio possuem o mesmo privilégio.

Os trabalhos propostos por MATTES e ZUFFO (2006) e PARK e HUMPHREY (2006), apesar de utilizarem tecnologias diferentes, trazem modelos de segurança parecidos. A Figura 13 apresenta um modelo semelhante aos propostos pelos dois trabalhos, no qual o usuário envia uma credencial de acesso contendo privilégios específicos à submissão da tarefa. Conforme os privilégios da credencial, configura-se um dispositivo do tipo máquina virtual, que é utilizado como ambiente de execução. Após sua inicialização, a máquina virtual limita a tarefa, de forma a ter acesso somente aos arquivos dos sistemas, endereços e portas *TCP/IP* que foram previstos.

Figura 13 – Modelo de controle de acesso semelhante aos propostos por MATTES e ZUFFO (2006) e PARK e HUMPHREY (2006).

A solução proposta por MATTES e ZUFFO (2006) faz uso da máquina virtual da plataforma Java. Já a solução proposta por PARK e HUMPHREY (2006) utiliza a

máquina virtual da plataforma “.net” (.NET, 2007). Para a geração de credenciais de acesso com privilégios de usuários, cada solução propõe sua versão modificada do serviço CAS do GT4.

3.6.3 Análise das Soluções de Sistema de Controle de Acesso para Grades Computacionais

Diversos trabalhos sobre controle de acesso em plataformas de grade computacional estão em andamento ou foram concluídos recentemente, demonstrando a relevância do tema e o interesse da comunidade acadêmica. Os trabalhos analisados seguem duas orientações distintas, uma voltada para infra-estruturas distribuídas e flexíveis de controle de acesso e outra para instanciação remota de tarefas.

Entre as infra-estruturas de controle de acesso para grades computacionais estudadas, os projetos CAS e Akenti utilizam linguagens flexíveis e certificados digitais para a efetivação de diferentes modelos de políticas. Já a infra-estrutura proposta por Lang et al. (2006) vai além, permitindo que, por meio do uso de interfaces de comunicação, se integrem políticas locais de um sítio ao processo de autorização. Estes trabalhos têm em comum uma limitação. Suas efetivações exigem o uso de aplicações confiáveis e desenvolvidas de forma específica para cada plataforma. Desta maneira, podem ser muito bem utilizadas nos serviços inerentes à grade computacional, oferecidos por aplicações próprias. Contudo, a sua utilização em ambientes de execução de tarefas não confiáveis exige maiores pesquisas.

Os trabalhos relacionados a controle de acesso de tarefas instanciadas remotamente têm como objetivo oferecer um ambiente de execução seguro, atuando justamente na área onde os sistemas acima não são aplicáveis. Entretanto, as soluções encontradas são pouco flexíveis e limitadas:

- Alguns trabalhos gerenciam contas e privilégios do sistema operacional, podendo contar até com modelos de políticas complexos como os existentes no projeto PRIMA. Contudo, por mais flexíveis que sejam as políticas adotadas, sua efetivação é realizada somente pelos mecanismos pré-existentes nos sistemas operacionais que não possuem capacidades

para efetivar regras complexas de controle de acesso;

- Outras pesquisas estudadas fazem uso de máquinas virtuais configuráveis para aprimorar a capacidade de controle sobre as aplicações. Apesar do avanço, tais soluções carecem ainda da flexibilidade exigida por um ambiente complexo e heterogêneo como o de uma grade computacional, não permitindo a efetivação de políticas com diferentes modelos e granularidades.

O GridMultiPolicy, quando comparado com os trabalhos estudados, se destaca por conter as seguintes características:

- Proporciona um sistema de autorização que pode ser utilizado tanto para controlar a ação de tarefas e serviços no ambiente de execução, como para controlar o oferecimento de serviços de rede. Como visto, nenhum dos trabalhos relacionados possui essa dupla capacidade.
- Oferece suporte a aplicações legadas, não exigindo alteração dos códigos-fontes das aplicações. Tal capacidade se faz presente somente nas soluções estudadas, baseadas em gerenciamento de contas do sistema operacional e em máquinas virtuais;
- Permite o estabelecimento dinâmico de diferentes níveis de granularidades de controle. Nenhum dos trabalhos estudados possui mecanismos semelhantes;
- Gerencia múltiplas políticas e mecanismos para a efetivação do controle de acesso. A proposta de Lang et al. (2006) faz uso de interfaces de comunicações locais para permitir a efetivação de múltiplas políticas. Contudo, não prevê como essas políticas serão armazenadas, estabelecidas e distribuídas.

3.7 Conclusões

Este capítulo apresentou a problemática da segurança em grades computacionais, traçando os requisitos necessários, mostrando de forma breve o padrão *OGSF* e comparando as soluções existentes.

Entre os trabalhos estudados, a *GT4* merece destaque, pois, além de seguir a padronização *OGSF*, é a plataforma que mais se coaduna com os requisitos de

segurança para grades computacionais abertas e padronizadas.

O GridMultiPolicy, comparado com os trabalhos estudados relativos a controle de acesso, contribui para a segurança de grades computacionais, principalmente em dois aspectos: oferece um serviço de controle de acesso flexível com suporte ao gerenciamento, efetivação e integração de múltiplas políticas independentes; e pode ser usado tanto no oferecimento de serviços de rede, como em ambiente de execução de tarefas.

Os resultados obtidos nesse capítulo são primordiais para entender o contexto e a necessidade da presente tese, que tem por objetivo, justamente, ajudar a preencher a lacuna relativa à segurança ainda existente nos ambientes de grade computacional.

4 Modelo GridMultiPolicy

Este capítulo apresenta o modelo de segurança GridMultiPolicy para gerenciamento e efetivação de políticas de controle de acesso de múltiplos formatos em ambientes de grades computacionais. A Seção 4.1 traz uma visão geral do modelo proposto e os pré-requisitos que nortearam seu desenvolvimento; a Seção 4.2 descreve o modelo de gerenciamento de políticas em múltiplos formatos; e a Seção 4.3 descreve o modelo de efetivação de políticas. Finalizando, algumas conclusões são apresentadas na Seção 4.4.

4.1 Pré-requisitos e Visão Geral

O GridMultiPolicy propõe um modelo de controle de acesso flexível e dinâmico para ser efetivado em aplicações instanciadas remotamente em grades computacionais. Entre as capacidades do modelo proposto, pode-se destacar a capacidade de estender o suporte a novas linguagens e modelos de políticas e estabelecer de forma dinâmica diferentes mecanismos de segurança.

A Figura 14 descreve o processo de autorização da plataforma Autenticação, Autorização e Auditoria (*Authentication, Authorization and Accounting – AAA*) (VOLLBRECHT et al., 2000), que serviu de base para a criação do GridMultiPolicy, no qual são previstos Ponto de Efetivação de Política (PEP), Ponto de Decisão de Política (PDP) e políticas com o objetivo de controlar o acesso das aplicações a um determinado recurso-alvo. Os PEPs são responsáveis por interceptar as “ações” das aplicações e consultar o PDP para a tomada de decisão. O PDP, por sua vez, baseia-se no contexto da requisição e nas políticas para negar ou autorizar a “ação”. Os PEPs estabelecidos oferecem suporte a uma determinada granularidade e o PDP deve ser capaz de decodificar e efetivar as políticas desenvolvidas em determinado padrão ou linguagem. Nesse sentido, o estabelecimento de um ambiente dinâmico de múltiplas políticas envolve o gerenciamento de PEPs e PDPs para que se possa atingir as granularidades e os suportes aos formatos de políticas desejados.

Figura 14 – Modelo de controle de acesso genérico da plataforma AAA.

Outro fator importante a ser abalizado na efetivação de múltiplas políticas é a integração de granularidades distintas. Dentro de um contexto de controle de acesso, um “alvo” é a representação da gama de recursos computacionais a ser protegida, podendo haver políticas com alvos e granularidades distintas, que remetem a recursos computacionais em comum. Tradicionalmente, políticas de granularidade distintas são efetivadas de forma independente, o que nem sempre atende aos requisitos necessários.

A

Figura 15 ilustra tal problemática, em que no sistema de segurança representado coexistem duas políticas de controle de acesso com diferentes níveis de granularidade. A primeira é relacionada aos tradicionais ler, modificar e apagar arquivos do sistema; a segunda controla o acesso a objetos em documentos *XML*. Para assegurar que determinados arquivos *XML* tenham acessos restringidos por uma granularidade de tal nível, as duas políticas devem trabalhar de forma integrada, impedindo que estes arquivos se tornem acessíveis sem o controle da política para *XML*.

Figura 15 – Problemática de diferentes níveis de granularidade.

A

Figura 16 traz uma visão do modelo GridMultiPolicy, que conta com um serviço para gerenciar políticas de segurança de múltiplos formatos e uma camada de efetivação de políticas.

Figura 16 – Modelo de segurança GridMultiPolicy.

No modelo proposto, cada política gerenciada é relacionada a um tipo de linguagem, PDP e nível de granularidade. Dessa forma, a camada de efetivação pode estabelecer corretamente o ambiente de execução de múltiplas políticas com

os PDPs para efetivação das políticas e os PEPs necessários aos níveis de granularidade previstos.

4.2 Modelo de Gerenciamento de Políticas de Múltiplos Formatos

A Figura 17 contém o modelo de gerenciamento de políticas de controle de acesso de múltiplos formatos para grades computacionais, cujo módulo principal coordena as regras gerais e pode ser auxiliado por módulos secundários, que utilizam linguagens próprias e oferecem regras direcionadas para um determinado escopo ou alvo. Os escopos são controlados pelo módulo principal e são utilizados para definir os contextos de atuação dos secundários.

Figura 17 – Modelo de gerenciamento de políticas em múltiplos formatos.

Os módulos secundários podem ser desenvolvidos por domínios autônomos e, para manter a interoperabilidade com o módulo principal, deve-se implementar uma interface de comunicação.

4.2.1 Escopo de utilização de políticas

A utilização de módulos secundários possibilita aos administradores, usuários e programadores estabelecerem, de forma independente, seus próprios modelos e linguagens de políticas. Contudo, a VO deve possuir meios para garantir que as políticas atuem somente nos recursos aos quais são destinadas. Com este intuito, os módulos secundários são associados a um escopo de atuação.

Os escopos são descritos primeiramente em relação a seu domínio lógico,

que inclui sítios e serviços e, em seguida, aos recursos computacionais envolvidos. Assim, um administrador de sítio pode definir suas próprias políticas e mecanismos de segurança, que serão válidos somente nos sítios em que é administrador. Já os usuários da VO podem estabelecer políticas e mecanismos para atuarem sobre os recursos computacionais que lhes pertence.

4.2.2 Informações para a Contextualização de Políticas

Além de oferecer regras de controle de acesso, o módulo principal relaciona as políticas com as informações úteis para o correto estabelecimento de ambientes destinados às suas efetivações. Estão previstas informações sobre políticas com duas finalidades:

- Estabelecimento dos PDPs: cada módulo secundário pode utilizar o seu próprio formato de política de segurança. Dessa forma, faz-se necessário o oferecimento de informações para que se possa estabelecer corretamente o PDP capaz de interpretar e fazer cumprir as políticas no formato adotado;
- Estabelecimento dos PEPs: no ambiente de execução faz-se necessário saber os níveis de granularidade envolvidos para que, na seqüência, se possam realizar as transformações necessárias ao estabelecimento dos PEPs previstos nos níveis de granularidade desejados.

4.2.3 Integração com Mecanismos Nativos

A segurança em redes computacionais apresenta um alto grau de evolução proveniente de décadas de pesquisa e investimentos, existindo atualmente uma infinidade de modelos e mecanismos de segurança. Dessa forma, a integração com mecanismos pré-existentes não é uma tarefa fácil. Exige conhecimentos sobre as tecnologias envolvidas e deve ser tratada caso a caso.

Dada a complexidade da problemática apresentada, o modelo proposto não se destina a oferecer uma resposta universal para integrações em segurança, mas, sim, um modelo flexível baseado em múltiplas políticas, que permita interoperar com os diferentes sistemas de segurança existentes.

Para apresentar uma idéia das suas capacidades, serão apresentadas a seguir duas metodologias que utilizam o modelo proposto para integração com mecanismos pré-existentes:

- Mapeamento e controle: metodologia que permite a integração com mecanismos de controle de acesso pré-existentes em serviços como banco de dados, servidor de arquivos etc. Consiste no estabelecimento de regras para o mapeamento em contas (*logins*) na utilização de um serviço e, paralelamente, no estabelecimento de políticas, PDP e PEPs específicos para exercer um controle sobre as requisições realizadas. Dessa maneira, a ação da aplicação pode ser tanto bloqueada pelas regras locais do serviço utilizado como pelo PDP estabelecido. A Figura 18 traz o privilégio efetivo no uso da metodologia descrita, a intersecção entre os privilégios da conta mapeada e os da política estabelecida pelo GridMultiPolicy.

Figura 18 – Privilégio efetivo, a intersecção entre os privilégios da conta mapeada e a da política estabelecida.

- Integração direta: metodologia que prevê a comunicação direta entre os PDPs de uma política com um determinado mecanismo do sítio. A comunicação estabelecida pode servir para oferecer detalhes sobre as operações do aplicativo ou para acatar suas decisões de autorização. Esta metodologia pode ser utilizada em conjunto com IDS, *Firewalls*, sistemas de controle de acesso etc.

4.2.4 Certificados de Políticas

Os certificados de políticas são documentos assinados digitalmente e são utilizados no modelo proposto para a distribuição de políticas de múltiplos formatos dentro de uma VO. A Figura 19 apresenta o modelo de certificado proposto, no qual além dos campos tradicionais de assegurado e assinatura digital, ainda se prevê o escopo principal que define o cenário de validade do certificado, as políticas gerais de acesso e, possivelmente, os escopos e as políticas associadas aos módulos secundários.

Figura 19 – Modelo de certificado de políticas.

As credenciais são criadas para um determinado contexto de segurança, que tem como base às características da tarefa ou serviço a serem submetidos, como sítio de hospedagem, proprietário, papel do usuário, propriedade do aplicativo utilizado e recursos necessários. Neste sentido, são utilizados os descritores de tarefas e as regras dos escopos, para determinar quais políticas devem ser incluídas em uma credencial.

4.2.5 Suporte ao Princípio de Privilégio Mínimo

O princípio de privilégio mínimo define que uma tarefa deva possuir somente acesso aos recursos necessários para seu cumprimento e não mais que isso. O modelo proposto possui três capacidades fundamentais para o desenvolvimento de políticas voltadas para o princípio de privilégio mínimo:

- Suporte ao estabelecimento de diferentes níveis de granularidade: para definir privilégios de forma mínima, faz-se necessário o emprego de granularidades específicas para o contexto dado;
- Permite o estabelecimento de políticas específicas para serviços e recursos: o princípio de privilégio mínimo é um conceito relativo. Sua efetivação depende da capacidade de desenvolver políticas específicas que respondam às necessidades dos recursos envolvidos;
- Segue um modelo de distribuição de políticas baseado em credenciais específicas aos contextos de utilização. Essa característica permite que os privilégios possam ser atribuídos minimamente para a realização de cada tarefa.

4.3 Modelo de Efetivação de Políticas

Segundo o modelo proposto, cada aplicação instanciada em sítio da grade computacional é gerenciada por um ambiente de execução configurado com os PEPs e PDPs específicos e as políticas de segurança estabelecidas para o contexto dado.

Na criação do ambiente de execução, o primeiro passo é o estabelecimento dos PEPs para que a camada de efetivação de políticas se posicione corretamente com a granularidade necessária entre a aplicação e os recursos, conforme exemplificado pela Figura 20. Com esse propósito, com base nos níveis de granularidade previstos, são realizadas transformações no ambiente de execução e nas bibliotecas envolvidas para o estabelecimento dos PEPs necessários.

Figura 20 – Posicionamento da camada de efetivação.

No processo de configuração do ambiente de execução, para cada política fornecida é estabelecido um PDP correspondente, responsável por sua efetivação. A Figura 21 traz a estrutura dos PDPs instanciados, na qual analogamente à estrutura de gerenciamento de políticas, o ambiente de execução é coordenado por um PDP principal, que pode contar com PDPs secundários para auxiliar no processo de tomada de decisões de acesso. Cada PDP secundário possui um escopo para determinar as condições onde seu uso é necessário e políticas a serem cumpridas, que são representadas segundo um determinado formato ou linguagem.

Figura 21 – Estrutura hierárquica do PDP.

Uma vez estabelecido o ambiente de execução, as ações das aplicações são interceptadas pelos PEPs e submetidas ao PDP principal para a tomada de decisão. Para cada requisição de autorização, o PDP principal coleta decisões dos PDPs secundários para que finalmente negue ou autorize a ação da aplicação.

4.4 Conclusões

Este capítulo apresentou o modelo GridMultiPolicy de controle de acesso para plataformas de grade computacional baseado em múltiplas políticas, composto por dois elementos integrados: um modelo de gerenciamento de políticas de múltiplos formatos e a camada de efetivação de políticas.

O modelo de gerenciamento de políticas de múltiplos formatos mostrou como promover uma integração de políticas com linguagens e níveis de granularidade distintos, utilizando, nesse sentido, uma estrutura hierárquica.

O modelo de efetivação de múltiplas políticas proposto prevê o estabelecimento de ambientes de execução configurados, com os PEPs e PDPs necessários ao estabelecimento dos níveis de granularidade previstos e à compreensão dos múltiplos formatos de políticas envolvidos.

A seguir, no Capítulo 5, apresentam-se com detalhes a arquitetura e os componentes da implementação do modelo proposto.

5 *GridMultiPolicy* – Arquitetura e Implementação

Este capítulo apresenta a arquitetura e implementação do sistema de controle de acesso *GridMultiPolicy* (MATTES; MILITELLI; ZUFFO, 2007a). A Seção 5.1 oferece uma visão geral da arquitetura *GridMultiPolicy*; a Seção 5.2 descreve o Ambiente Java de Múltiplas Políticas (*Java Multi Policy Environment – JMPE*) e a Seção 5.3 apresenta o Serviço de Gerenciamento de Múltiplas Políticas (*Multi Policy Manager Service – MPMS*).

5.1 *Arquitetura GridMultiPolicy* - Visão Geral

O *GridMultiPolicy* é composto por dois elementos: o *MPMS*, responsável por gerenciar e distribuir políticas de múltiplos formatos; e o *JMPE*, incumbido de prover um ambiente para a efetivação das políticas estabelecidas pelo *MPMS*. Estes componentes são representados na Figura 22, na qual se observa que, para cada tarefa ou serviço instanciado no sítio, é estabelecida uma instância configurada do *JMPE*, como os PEPs, PDPs e as políticas previstas para o contexto dado.

Figura 22 – Arquitetura geral do GridMultiPolicy.

Cada instância do *JMPE*, localizada entre uma aplicação e o sistema operacional, exerce controle sobre as ações que a tarefa ou serviço realiza no ambiente de execução. Adicionalmente, o *JMPE* pode ter políticas para controlar o oferecimento do serviço de rede para usuários da VO.

5.2 JMPE

O *JMPE* é a entidade responsável pelo estabelecimento de um ambiente de execução, configurado para oferecer suporte às múltiplas políticas previstas. A efetivação de políticas de autorização é realizada pelo trabalho em conjunto de PEPs e PDPs, no qual os PEPs, situados estrategicamente entre a aplicação e os recursos, são responsáveis por interceptar as ações das aplicações e consultar os PDPs, que realizam as decisões de controle de acesso. Dessa maneira, o processo de efetivação de políticas compreende o estabelecimento correto dos PEPs e PDPs necessários.

5.2.1 Estabelecimento dos PDPs

O *MPMS* contém definições e escopos de atuação das múltiplas políticas que gerencia. Estas informações são disponibilizadas em certificados digitais, como o representado na Figura 23. Desta maneira, por meio do uso dos certificados emitidos pelo *MPMS*, o *JMPE* é capaz de estabelecer ambientes para efetivar as múltiplas políticas estabelecidas.

A Figura 23 exemplifica o processo em que, para um dado contexto de execução, e com base nas definições das políticas de uma *VO*, é estabelecido um *JMPE* configurado, contendo um PDP principal, auxiliado por múltiplos PDPs secundários. Para estabelecer quais PDPs devem ser utilizados, são analisados os escopos das políticas, que contêm regras descritas em *XACML*. Quando aplicadas em um contexto de execução, tais regras definem quais as políticas necessárias. A Seção 5.3.3 descreve escopos de políticas.

Figura 23 – Estabelecimento de ambiente de execução configurado.

5.2.2 Estabelecimento dos PEPs

Os PEPs são utilizados para oferecer controle a um determinado nível de granularidade. O *JMPE* contém PEPs preestabelecidos para uma granularidade básica e permite que se estabeleçam dinamicamente novos PEPs para outros níveis de granularidade.

Granularidade Básica *JMPE*

Em condições normais, as aplicações Java não possuem um gerente de segurança, sendo suas operações sobre os recursos disponíveis controladas pelo sistema operacional. No entanto, existem meios de atribuir a uma aplicação um gerente de segurança, que passa a controlar as ações por meio de uma série de pontos de checagem existentes na plataforma Java (PISTOIA et al., 1999). A Figura 24 contém um trecho de código no qual se verifica a permissão para finalizar a execução da aplicação (HERZOG; SHAHMEHRI, 2001). Vale observar que, caso o gerente de segurança não seja instanciado, o método “System.GetSecurityManager()” da Figura 24 irá retornar nulo e, deste modo, nenhuma checagem será realizada. Caso contrário, a checagem será executada pelo gerente que gera uma exceção de segurança para cada tentativa de ação não permitida.

```
1  SecurityManager security = System.getSecurityManager();  
2  if (security != null){  
3      security.checkExit(status); }
```

Figura 24 – Ponto de efetivação de política da plataforma Java.

Para o estabelecimento dos PEPs da granularidade básica, o *JMPE* conta com a classe “br.usp.lsi.BasicPEP”, que é uma extensão do gerente de segurança Java padrão “java.lang.SecurityManager”. Desta maneira, a granularidade básica herdou os PEPs existentes no gerente de segurança Java, o que lhe permite controlar as ações de ler, escrever e apagar nos arquivos do sistema e de estabelecer conexões *TCP/IP*.

Estabelecimento dos PEPs de Outros Níveis de Granularidade

Voltando a analisar a Figura 15 da Seção 4.1, verificamos também que os PEPs são estabelecidos dentro do contexto de uma granularidade, exemplificando: PEPs de arquivo ocorrem no contexto de acesso a arquivos; PEPs de *XML*, no contexto de operações *XML*; PEPs de banco de dados, no contexto de acesso a banco de dados etc. Desta maneira, o grande desafio para o estabelecimento de um nível de granularidade é “entrar” no respectivo contexto para a criação dos PEPs necessários.

O GridMultiPolicy prevê a realização de modificações em bibliotecas, no intuito de estabelecer os PEPs necessários de um dado contexto. Para exemplificar, temos o estabelecimento de um PEP para uma granularidade de serviços web por meio de modificações na biblioteca Axis (AXIS, 2007). A Figura 25 traz duas versões do código-fonte do método “invoke”: a) original sem modificações; b) depois das modificações, que inserem o PEP responsável por coletar as informações de contexto (nas linha 3, 4 e 5 são coletado os parâmetros “*host*”, “*service*” e “*port*”) e consultar o *JMPE* sobre decisões de acesso (linha 6). Desta maneira, após o PEP inserido, quando da utilização da biblioteca Axis para acessar serviços web, o *JMPE* é consultado e, caso necessário, pode evitar a ação, gerando uma exceção de segurança.

Figura 25 – Código do método "invoke()" da classe org.apache.axis.client.Call a) original sem modificações b) após modificações que incluem o PEP para granularidade de serviços web.

Para a melhor compressão do mecanismo *JMPE* de estabelecimento dos PEPs, faz-se necessário apresentar, primeiramente, o processo de carregamento de classes em tempo de execução na plataforma Java e, em segundo lugar, a biblioteca Javassist (JAVAASSIST, 2007) para manipulação de classes Java compiladas.

Na plataforma Java, as classes são armazenadas em arquivos binários contendo *bytecodes* Java e em tempo de execução são carregadas para serem interpretadas pela Máquina Virtual Java (*Java Virtual Machine - JVM*). O processo para carregar classes é realizado por um ou mais objetos do tipo "ClassLoader". Cada objeto "ClassLoader" possui métodos predefinidos, que têm como função, a partir do nome da classe, encontrar o arquivo correspondente, lê-lo e retornar os

bytecodes da classe para *JVM*. A plataforma Java oferece meios para que se construam e utilizem novos “ClassLoaders” com características distintas (Java, 2006).

A biblioteca Javassist oferece uma interface para a manipulação de *bytecodes* Java, permitindo que, a partir do arquivo binário, seja possível alterar, adicionar e/ou apagar métodos, propriedades e interfaces de uma classe compilada.

O *JMPE* conta com a classe “br.lsi.JMPEClassLoader” para carregar e modificar classes em tempo de execução, com o objetivo de inserir os PEPs necessários. As modificações a serem realizadas são passadas por meio de instruções *XML*, que definem os métodos e as classes a serem alteradas para inserção dos PEPs.

A Figura 26 traz como exemplo a instrução para inserção de um PEP para controlar o uso de serviços web na biblioteca Axis. As linhas 8 e 9 definem a classe e o método em que o PEP deverá ser inserido. A linha 5 especifica o “webServiceInvoke” como nome da ação a ser autorizada. Adicionalmente, as linhas 10, 11 e 12 determinam que o PEP a ser inserido deve passar os parâmetros “host”, “service” e “port” para o *JMPE*. Tais parâmetros se tornaram acessíveis aos PDPs para auxiliar na tomada de decisão.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <ActionInstruction xmlns="lsi.usp.br/PEPPolicy.xsd"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="lsi.usp.br/PEPPolicy.xsd PEPPolicy.xsd" >
5      <ActionName>webServiceInvoke</ActionName>
6      <PEP>
7          <TypeTarget>Class</TypeTarget>
8          <ClassName>org.apache.axis.client.Call</ClassName>
9          <MethodName>invoke</MethodName>
10         <PropertyName>host:transport.url</PropertyName>
11         <PropertyName>service:operationName.getNamespaceURI()</PropertyName>
12         <PropertyName>port:operationName.getLocalPart()</PropertyName>
13     </PEP>
14 </ActionInstruction>

```

Figura 26 – Instrução para criação do PEP na biblioteca Axis.

A utilização do “br.lsi.JMPEClassLoader” envolve dois momentos:

- Primeiro, no estabelecimento do ambiente de execução, quando, com base nas ações descritas nos escopos da credencial de acesso, é fornecida uma lista de instruções *XML* para “br.lsi.JMPEClassLoader”

(passo 0 da Figura 27);

- No segundo, em tempo de execução, as classes são carregadas e modificadas (passos de 1 a 5 da Figura 27).

A Figura 27 contém o processo em que as classes são carregadas e modificadas, no qual o “br.lsi.JMPEClassLoader” procede por meio dos seguintes passos:

1. Recebe da *JVM* uma requisição para carregar uma determinada classe;
2. Carrega do sistema do arquivo os *bytecodes* associados ao nome fornecido da classe;
3. Verifica a existência de instruções associadas à classe carregada;
4. Utiliza, caso necessário, a biblioteca JavaAssist para manipular os *bytecodes* e estabelecer os PEPs previstos;
5. Retorna para a *JVM* os *bytecodes* da classe possivelmente alterados.

Figura 27 – Efetivação das Políticas de Segurança

A função principal do *JMPE* é a efetivação das políticas de segurança em diferentes formatos. Como descrito na Figura 23 o *JMPE* possui o PDP principal para coordenar o processo de tomada de decisão e pode contar com diversos PDPs secundários para auxiliá-lo, sendo que cada PDP secundário é limitado por um escopo de atuação e oferece suporte a um determinado formato de política.

A comunicação entre o PDP principal e os secundários ocorre por meio da interface “br.usp.lsi.JMPE.PDPInterface”, implementada por todos os PDPs. A Figura 28 mostra o código da interface “br.usp.lsi.JMPE.PDPInterface”, no qual os

métodos “*permitFile*” e “*permitSock*” são utilizados respectivamente para decisões de acesso a arquivos locais e conexões de rede; o “*permiLoadLib*” é utilizado para autorizar o acesso às bibliotecas do sistema operacional e o “*permitOperation*” é genérico, sendo utilizado para autorização das demais ações. Os métodos “*initSession*” e “*destroySession*” são utilizados no controle de acesso em serviços de rede para que as políticas possam criar contextos específicos aos usuários da VO que venham a realizar requisições.

```

1 public interface PDEInterface {
2     public PDPAuthorization permitFile(String filename
3         ,int operation, Object executionContext) ;
4     public PDPAuthorization permitSock(String host,
5         int port, int operation, Object context) ;
6     public PDPAuthorization permiOperation(String operation,
7         Object[] parameters) ;
8     public PDPAuthorization permiLoadLib(String lib) ;
9     public void initSession(Object[] parameters);
10    public void DestroySession(Object[] parameters);
11 }

```

Figura 28 – Interface de comunicação PDP “br.usp.isi. JMPE.PDEInterface”.

A cada requisição, um PDE retorna um objeto do tipo *PDPAuthorization* que pode assumir cinco valores possíveis — “Permitido”, “Negado”, “Indeterminado”, “Não aplicável” e “Restrito”. O valor “Restrito” é utilizado para integração de políticas que têm como alvo os mesmos recursos, porém, com ações de granularidade distintas. Assim, quando uma política retorna “restrita”, informa que a ação deve ser somente permitida caso o recurso esteja sendo controlado em uma granularidade mais refinada.

A cada requisição de autorização recebida dos PEPs, o PDP principal prossegue por meio dos seguintes passos:

1. Coleta as decisões dos PDPs secundários para a dada ação;
2. Aplica, caso necessário, o algoritmo de resolução de conflito;
3. Autoriza ou nega a ação.

Para a resolução de conflitos é permitido o uso de três algoritmos fornecidos pela especificação *XACML* (XACML, 2007): Preferência pela Negação (*deny-overrides*); Preferência pela Permissão (*allows-overrides*); Primeiro Aplicável, (*first-applicable*).

A Figura 29 traz o pseudocódigo do algoritmo Preferência pela Negação no

qual, em caso de conflito, a negação de autorização se sobrepõe a outras decisões. Na utilização do algoritmo Preferência pela Negação, se uma única política negar, a ação não será realizada.

```

1  Decision denyOverridesPolicyCombiningAlgorithm(Policy policy[]) {
2      Boolean atLeastOnePermit = false;
3      for( i=0 ; i < lengthOf(policy) ; i++ ) {
4          Decision decision = evaluate(policy[i]);
5          if (decision == Deny) {
6              return Deny;
7          }
8          if (decision == Permit) {
9              atLeastOnePermit = true;
10             continue;
11         }
12         if (decision == NotApplicable) {
13             continue;
14         }
15         if (decision == Indeterminate) {
16             return Deny;
17         }
18     }
19     if (atLeastOnePermit) {
20         return Permit;
21     }
22     return NotApplicable;
23 }

```

Figura 29 – Algoritmo de resolução de conflito Preferência pela Negação (XACML 2007).

A Figura 30 contém o pseudocódigo do algoritmo Preferência pela Permissão no qual, em caso de conflito, a autorização se sobrepõe às outras decisões. No emprego do algoritmo Preferência pela Permissão, basta uma das políticas permitir para a ação ser autorizada.


```

1  Decision permitOverridesPolicyCombiningAlgorithm(Policy policy[]) {
2      Boolean atLeastOneError = false;
3      Boolean atLeastOneDeny = false;
4      for( i=0 ; i < lengthOf(policy) ; i++ ) {
5          Decision decision = evaluate(policy[i]);
6          if (decision == Deny) {
7              atLeastOneDeny = true;
8              continue;
9          }
10         if (decision == Permit){
11             return Permit;}
12         if (decision == NotApplicable) {
13             continue;}
14         if (decision == Indeterminate) {
15             atLeastOneError = true;
16             continue;}
17     }
18     if (atLeastOneDeny) {
19         return Deny;}
20     if (atLeastOneError) {
21         return Indeterminate;}
22     return NotApplicable;
23 }

```

Figura 30 – Algoritmo de resolução de conflito Preferência pela Autorização (XACML 2007).

A Figura 31 traz o pseudocódigo do algoritmo Primeiro Aplicável, no qual se estabelece uma ordem de prioridade entre as políticas e, em caso de conflito, a primeira política que possuir regra para a ação é utilizada.

```

1  Decision firstApplicableEffectPolicyCombiningAlgorithm
2      (Policy policy[]) {
3      for( i = 0 ; i < lengthOf(policy) ; i++ ) {
4          Decision decision = evaluate(policy[i]);
5          if(decision == Deny) {
6              return Deny;}
7          if(decision == Permit) {
8              return Permit;}
9          if (decision == NotApplicable) {
10             continue;}
11         if (decision == Indeterminate) {
12             return Indeterminate;}
13     }
14     return NotApplicable;
15 }

```

Figura 31 – Algoritmo de resolução de conflito Primeiro Aplicável (XACML 2007).

5.2.3 Integrando Níveis de Granularidade

Como já discutido, a efetivação de um nível de granularidade maior depende de sua integração com o nível menor. Tomemos o exemplo da Seção 5.2.2: ao inserir o PEP na classe da AXIS (AXIS, 2007), o *JMPE* permite que o controle de acesso possa ser exercido no nível do serviço web. Contudo, a aplicação pode utilizar serviços web por meio de outras bibliotecas como *JAX-WS* (JAX-WS, 2007) ou, até mesmo, construir suas próprias classes de acesso, possibilitando que se faça uso de recursos sem o controle previsto. Para evitar tal situação, devem-se fornecer regras para integrar o controle de acesso dos serviços web com o *TCP/IP* (nível de granularidade menor), garantindo que, em determinados contextos, a aplicação somente consiga utilizar recursos da rede por meio de classes com controle embutido.

Regras *XACML* do tipo *Restrict*, descrita na Seção 5.3.2, são propostas pelo presente trabalho, no intuito de promover a integração de ações com níveis de granularidade hierarquicamente ligados, tendo como alvo situações como o exemplo acima sugerido, no qual ações de serviços web com granularidade maior são relacionadas às ações *TCP/IP* de granularidade menor.

Na fase de instanciação da aplicação, o *JMPE* guarda duas listas: a primeira, descrita na Seção 5.2.2, contém instruções para inserção de PEPs em classes para a autorização de ações; a segunda lista relaciona ações de granularidade maior com ações de granularidade menor. Em tempo de execução, as duas listas são utilizadas para a efetivação de regras '*Restrict*'. Quando o *JMPE* recebe *Restrict* como resposta de um PDP, prossegue por meio dos seguintes passos (Figura 32):

1. A partir da ação original e dos escopos de políticas, seleciona as ações de menor granularidade correlacionadas;
2. Com base na seleção do passo 1, e na lista de instruções de PEP, estabelece a lista de classes que foram alteradas para inserir os PEPs com ações de granularidade maior;
3. Se a requisição da ação original é proveniente de uma das classes Java listada no passo anterior, transforma a resposta do PEP em *Permit*; caso contrário, em *NotApplicable*.

Figura 32 – Efetivação de uma regra ‘Restrict’.

5.3 MPMS

A Seção 5.3.1 mostra a visão geral do Serviço de Gerenciamento de Múltiplas Políticas (*Multi Policy Manager Service - MPMS*); a Seção 5.3.2 apresenta a extensão da linguagem XACML produzida para oferecer suporte às necessidades do *MPMS*; a Seção 5.3.3 traz o serviço de gerenciamento de políticas, e a Seção 5.3.4 apresenta o serviço de credencial de acesso.

5.3.1 MPMS Visão Geral

O *MPMS* é responsável por administrar, armazenar e distribuir políticas de segurança de múltiplos formatos em uma *VO*. A estrutura do *MPMS* segue o modelo apresentado na Seção 4.2, contando com um módulo de política principal para fornecer as regras gerais de acesso e gerenciar os módulos de políticas secundários. As políticas do módulo principal são gerenciadas por meio de uma extensão da linguagem XACML.

5.3.2 Extensão da Linguagem *XACML*

Esta seção apresenta as modificações realizadas na linguagem *XACML* para que pudesse atender às necessidades do GridMultiPolicy em relação à linguagem de políticas.

A linguagem *XACML* foi estabelecida com base na lógica apresentada na Seção 2.2.4, e permite a representação de diferentes políticas de controle de acesso. Na *XACML* toda política é estabelecida para controlar o acesso a um determinado alvo, entidade que representa um dado conjunto de recursos sobre os quais podem ser realizadas ações que devem ser controladas.

Como já foi discutido na Seção 4.1, um determinado alvo ou gama de recursos podem ser associados a ações de granularidades distintas e um controle de acessos eficiente deve ser capaz de promover regras que integram os diferentes níveis de granularidade. Deste modo, na primeira modificação realizada no *XACML* foi acrescentada a capacidade de representar o relacionamento entre ações com níveis de granularidade hierárquicos correlacionados. A Figura 33 representa o modelo de linguagem proposto que se diferencia do *XACML* original somente pelo relacionamento “tracejado”, que liga as ações de granularidade menor com as de maior. Exemplificando: a granularidade “*XML*” é relacionada à de “arquivos”, pois, para ter acesso às operações de *XML*, primeiro é necessário ter acesso ao arquivo do sistema.

Figura 33 - Modelo da linguagem de política proposto.

Além de fornecer uma linguagem para representar políticas de autorização, o XACML especifica o modelo de autorização descrito pela Figura 2 da Seção 2.2.4 — no qual o PEP é responsável por interceptar as requisições de acesso e repassá-las ao PDP. Como decisão de acesso, o PDP oferece como resposta um objeto *Decision* (decisão).

O XACML original previa quatro valores possíveis para objetos *Decision*: *Permit* (Permitido), *Deny* (Negado), *Indeterminate* (Indeterminado), *NotApplicable* (não aplicável). No intuito de oferecer regras para integração entre níveis e granularidade, foi adicionado mais um valor possível para o objeto *Decision*: *Restrict* (Restrito), que é utilizado para explicitar que dada ação deve ser autorizada, se, e somente se, existir outra política que controle o recurso em uma granularidade maior. A Figura 34 traz o elemento “*Decision*” do esquema XACML modificado.

```

1  <!-- -->
2  <xs:element name="Decision"
3      type="xacml-context:DecisionType"/>
4  <xs:simpleType name="DecisionType">
5      <xs:restriction base="xs:string">
6          <xs:enumeration value="Permit"/>
7          <xs:enumeration value="Deny"/>
8          <xs:enumeration value="Indeterminate"/>
9          <xs:enumeration value="NotApplicable"/>
10         <xs:enumeration value="Restrict"/>
11     </xs:restriction>
12 </xs:simpleType>
13 <!-- -->

```

Figura 34 - Elemento “Decision” do XACML modificado.

5.3.3 Gerenciamento de Privilégios

O módulo principal de política do *MPMS* é o responsável pelo gerenciamento das políticas gerais da *VO* e dos módulos secundários. O emprego da linguagem *XACML* atribui flexibilidade ao módulo principal, permitindo-lhe estabelecer diferentes modelos de políticas, níveis de granularidade e regras.

O uso de módulos secundários permite que novos tipos de políticas possam ser utilizados para auxiliar o controle de acesso em determinados sítios, serviços e/ou recursos. Contudo, como os módulos secundários adotam seu próprio formato de regras, é necessário o emprego de escopos de atuação para permitir que o módulo principal, mesmo sem conhecer os formatos de política envolvidos, consiga delimitar a atuação das políticas secundárias. Desta maneira, todo módulo secundário é associado a um escopo de atuação.

Os escopos de atuação são codificados em *XACML* e definem os seguintes parâmetros de políticas:

- O alvo que a política é destinada a proteger;
- O PDP responsável por sua efetivação;
- As ações previstas.

As regras de escopo são processadas, em primeira instância, no *MPMS*, para gerar um certificado contendo as definições de políticas para um sítio. São utilizadas também no processo de estabelecimento de *JMPE* configurados, onde se determina

a política necessária a um contexto de utilização.

Para processar as regras de escopos, os sítios fazem uso dos parâmetros encontrados nos certificados dos usuários responsáveis pela submissão de uma tarefa ou utilização de um serviço, além de arquivos no formato *RSL*, que descrevem tarefas a serem processadas.

A Figura 35 traz, a título de exemplo, parte do escopo da política “webservice”. Na primeira linha, são definidos o nome da política e a classe do PEP para sua efetivação. A linha 8 define qual política é destinada a proteger serviços web, e a linha 20 define “WebServiceInvoke” como ação sobre a qual a política deve exercer controle.

```

1  <Policy PolicyId="webServiceControl" PDPClass= "PDFWebService"
2      PDPJAR="PDPWeb.jar">
3      <Resources>
4          <Resource>
5              <ResourceMatch
6                  MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
7                      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
8                          urn:service:testeWeb</AttributeValue>
9                      <ResourceAttributeDesignator
10                         DataType="http://www.w3.org/2001/XMLSchema#anyURI"
11                         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
12                  </ResourceMatch>
13          </Resource>
14      </Resources>
15      <Actions>
16          <Action>
17              <ActionMatch
18                  MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
19                      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
20                          webServiceInvoke</AttributeValue>
21                      <ActionAttributeDesignator
22                         DataType="http://www.w3.org/2001/XMLSchema#string"
23                         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
24                  </ActionMatch>
25          </Action>
26      </Actions>
27  </Target>

```

Figura 35 - Escopo da política “webservice”.

A Figura 36 mostra outro exemplo de regra de escopo na qual, na linha 5, se define uma condição em que o parâmetro “*RSL:attribute:Domain*” (linha 11) deve ser igual a “*Isi.usp.br*” (linha 8). Desta maneira, as políticas que tiverem essa regra no escopo só deverão ser utilizadas para controlar tarefas e serviços instanciados em sítios do domínio “*Isi.usp.br*”.

```

1  <Target>
2  <Subjects>
3  <Subject>
4      <SubjectMatch MatchId="
5          urn:oasis:names:tc:xacml:1.0:function:string-equal">
6          <AttributeValue
7              DataType="http://www.w3.org/2001/XMLSchema#string">
8              lsi.usp.br
9          </AttributeValue>
10         <SubjectAttributeDesignator
11             AttributeId="urn:oasis:names:tc:xacml:2.0:RSL:attribute:Domain"
12             DataType="http://www.w3.org/2001/XMLSchema#string"/>
13         </SubjectMatch>
14     </Subject>
15 </Subjects>

```

Figura 36 - Regra de escopo que define o uso de políticas em sítios do domínio “lsi.usp.br”

A Figura 37 traz outro exemplo de regra de escopo na qual, na linha 5, se define uma condição em que o parâmetro “*RSL:attribute:required_resource*” (linha 11) deve ser igual a “*DataBase_Service*” (linha 8). Desta maneira, as políticas que tiverem essa regra no escopo só deverão ser utilizadas para controlar tarefas e serviços que requisitarem o uso de banco de dados.

```

1  <Target>
2  <Subjects>
3  <Subject>
4      <SubjectMatch MatchId="
5          urn:oasis:names:tc:xacml:1.0:function:string-equal">
6          <AttributeValue
7              DataType="http://www.w3.org/2001/XMLSchema#string">
8              DataBase_service
9          </AttributeValue>
10         <SubjectAttributeDesignator
11             AttributeId="urn:oasis:names:tc:xacml:2.0:
12             RSL:attribute:Required_resource"
13             DataType="http://www.w3.org/2001/XMLSchema#string"/>
14         </SubjectMatch>
15     </Subject>
16 </Subjects>
17 ...

```

Figura 37 - Regra de escopo que define o uso de políticas para controlar tarefas e serviços que requisitarem o uso de banco de dados.

5.3.4 Distribuição de Políticas

A cada contexto de execução, os PDPs do *JMPE* necessitam de políticas específicas para corretamente tomar as decisões de autorização. A Figura 38 traz o processo do serviço de distribuição de múltiplas políticas, que recebe como parâmetros um arquivo *RSL* e a credencial de acesso do usuário. O arquivo *RSL* descreve a tarefa ou serviço a ser executado. Após consultar os diferentes módulos secundários de políticas envolvidas, o serviço retorna políticas de múltiplos formatos, específicas para o contexto de execução.

Figura 38 – Serviço de distribuição de políticas de múltiplos formato.

A comunicação entre o módulo principal e os secundários ocorre por meio da interface “*PolicyInterface*” da Figura 39, que é implementada por todos os módulos. O método “*CreatePolicy*” define o processo de obtenção de políticas específicas para contextos de execução, cujo parâmetro “*Description*” é referente ao arquivo *RSL*. O parâmetro “*user*” é referente ao certificado digital X.509 do usuário. Assim, o *MPMS* consegue utilizar seus múltiplos módulos para a realização de políticas de diferentes formatos, contendo regras específicas para contextos de utilização. 3

```

1 public interface PolicyInterface {
2     public byte[] createPolicy(RSL description,
3         Certificate user);
4 }

```

Figura 39 - Interface “*PolicyInterface*” para a comunicação entre o módulo principal e os secundários, na geração da credencial de acesso.

5.4 Integrando GridMultiPolicy ao GT4

O GridMultiPolicy foi concebido para ser utilizado por diversas plataformas de grade computacionais. No intuito de oferecer uma visão prática de suas capacidades, esta seção apresenta como o GridMultiPolicy foi integrado aos serviços *WS Gram* e *Java WS Core* do *Globus Tool Kit*, versão 4.2 (GT4).

O serviço *WS Gram* do *GT4* possibilita que usuários de uma *VO* instanciem tarefas remotamente. O componente *Java WS Core* (Java WS Core, 2007) oferece uma infra-estrutura para hospedar serviços web com informações de estados (*stateful*).

5.4.1 Integrando GridMultiPolicy ao WS Gram

A Figura 40 mostra o processo de submissão de tarefas no *WS GRAM* integrado ao GridMultiPolicy, que contempla os seguintes passos:

1. O usuário envia uma requisição para submissão de tarefas, contendo um arquivo *RSL* e seu certificado X.509, para autenticação;
2. O *WS GRAM* envia um requisição para o *JMPE* que cria um ambiente de execução configurado com os PDPs e PEPs previstos;
3. Caso seja necessário, os PDPs obtêm políticas no *MPMS* ;
4. A tarefa é iniciada.

Figura 40 – Processo de submissão de tarefas.

5.4.2 Integrando GridMultiPolicy ao Java WS Core

Antes de discutir a integração com o *Java WS Core*, faz-se necessário introduzir alguns conceitos básicos:

- Contêiner é um módulo do *Java WS Core* que contém serviços, com suas próprias bibliotecas e ambiente de execução;
- Arquivos no formato GAR são utilizados para empacotar bibliotecas, arquivos e instruções para a criação de um contêiner.

A Figura 41 mostra as etapas do processo de submissão de serviços no *Java WS Core* integrado ao GridMultiPolicy:

1. O usuário envia uma requisição para submissão de serviço, contendo um arquivo GAR, a descrição do serviço no formato *RSL* e o certificado X.509 do usuário;
2. O *JMPE* cria um ambiente de execução configurado para a instanciação do contêiner;
3. Caso necessário, novas políticas são obtidas no *MPMS*;
4. O contêiner é inicializado.

Figura 41 – Processo de submissão de serviços.

O uso do GridMultiPolicy no *Java WS Core* permite a integração de políticas de controle de acesso que podem ter dois objetivos: controlar as ações que o serviço realiza e controlar o acesso dos usuários da *VO* no serviço instanciado.

No modelo adotado, cada contêiner possui uma única instância configurada do *JMPE*. Para controlar o acesso dos usuários da *VO* ao serviço, o *JMPE* faz uso dos métodos “*initSession*” e “*destroySession*” da Figura 28. Deste modo, permite que os PDPs utilizem os contextos de usuários gerados pelo *Java WS Core* para criar seus próprios contextos de segurança e controlar as requisições, conforme as características e privilégios dos usuários.

A Figura 42 apresenta o processo de autorização no oferecimento de serviços pelo *Java WS Core*, que é viabilizado conforme as seguintes etapas:

1. O usuário envia uma requisição de serviço contendo sua credencial de acesso;

2. O *JMPE* envia uma mensagem contendo as informações dos usuários para que os PDPs possam criar o seu contexto de segurança;
3. As requisições do usuário são processadas e controladas pelos PDPs com suas respectivas políticas e contextos de segurança;
4. As respostas das requisições são enviadas ao usuário.

Figura 42 – Processo de autorização no Java WS Core.

5.5 Conclusões

Este capítulo apresentou a arquitetura do sistema de autorização GridMultiPolicy e sua integração com a infra-estrutura de grade computacional *GT4*. O sistema GridMultiPolicy é composto de dois elementos integrados: o *MPMS*, capaz de gerenciar políticas em múltiplos formatos ou linguagens, e o *JMPE* responsável por efetivar as políticas. O sistema proposto permite o estabelecimento de políticas para serem utilizadas para exercer controle sobre o uso de determinados recursos-alvos. Em um contexto de execução de tarefa ou serviço, é realizado um ambiente configurado, capaz de cumprir as políticas estabelecidas. A inserção dos PEPs necessários em bibliotecas permite a especificação de diferentes níveis de granularidade, sem exigir a reformulação dos aplicativos.

O próximo capítulo demonstra como o sistema proposto pode oferecer respostas coerentes a desafios de segurança existentes em diferentes ambientes e aplicações.

6 Cenários de Uso e Estudo de Caso

O GridMultiPolicy foi concebido para ser um sistema de controle de acesso coerente com as necessidades de uma grade computacional, oferecendo um serviço flexível e integrado sobre uma plataforma heterogênea. Seu uso pode propiciar respostas a diferentes contextos e requisitos presentes nos sistemas envolvidos. Este capítulo analisa possíveis cenários de utilização de uma grade computacional, apontando as necessidades de segurança envolvidas. Posteriormente, no intuito de mostrar as capacidades do sistema proposto, são desenvolvidas políticas que oferecem respostas aos requisitos elencados. O capítulo aborda ainda um estudo de caso, com uma visão mais prática dos benefícios que o uso do sistema pode oferecer.

6.1 Cenário C1 – Instanciação de Tarefas e Serviços em Sítios com Mecanismos de Proteção

Uma determinada corporação, convencida dos benéficos que o uso da grade computacional pode trazer, resolve entrar em uma VO. A rede da corporação possui mecanismos específicos para proteger os seus recursos. A Figura 43 representa a arquitetura da rede em questão, na qual, o mecanismo do tipo *firewall* analisa o tráfego para limitar os serviços disponibilizados na Internet, e o Sistema de Detecção e Prevenção de Intrusos (*Intrusion Detection and Prevention System - IDS/IPS*) identifica e bloqueia conexões e sítios envolvidos em ataques. Para a detecção de ataques, o sistema de *IDS/IPS* conta com agentes espalhados na rede que analisam o tráfego ou informações nos servidores, identificando padrões suspeitos (DEBAR; DACIER; WESPI, 1999).

Figura 43 - Rede corporativa com sistema IDS distribuído.

Ao permitir que por meio da infra-estrutura da grade computacional, outras instituições instanciem remotamente tarefas e serviços, além de garantir maior cooperação entre membros da VO, novos desafios de segurança são adicionados. As tarefas instanciadas remotamente podem realizar ataques de dentro da rede e os mecanismos de proteção existentes, nesse novo contexto, perdem sua efetividade. O mecanismo analisador de pacotes é uma ferramenta de perímetro que bloqueia ataques provenientes de sítios externos, de modo que não é efetivo em aplicações instanciadas dentro da rede. O serviço de *IDS/IPS*, ao detectar um ataque, identifica o sítio proveniente. No caso de ataques realizados por tarefas e serviços instanciados internamente, o *IDS/IPS* apontará como atacante um sítio da rede que contém diversas aplicações sendo executadas simultaneamente, tornando-se difícil identificar entre muitas, qual é a responsável pelas ações suspeitas.

Um serviço de submissão de tarefas e/ou serviços para essa rede deveria integrar o ambiente de execução com os mecanismos de segurança utilizados (*IDS/IPS*, analisador de pacotes, *Firewall* e *Proxy*). Dessa forma, a utilização das tecnologias de grade computacional, não impactaria negativamente na segurança do sistema, não sendo necessária a reformulação completa dos sistemas de segurança existentes.

A integração com mecanismos pré-existentes não é uma tarefa fácil e depende das características específicas dos componentes empregados. O GridMultiPolicy não oferece uma solução geral para tal problema. Contudo, ao permitir a criação dinâmica de PEPs e PDPs, possibilita a implementação e a

efetivação de políticas que tenham como objetivo a interligação do controle de acesso a outros mecanismos. As Seções 6.5.1 e 6.5.2 trazem dois modelos de políticas que realizam este papel: o LeastPrivilege, que oferece funcionalidades similares aos mecanismos do tipo filtro de pacotes; e o IDS_Control, que interliga o ambiente de execução das aplicações com um mecanismo do tipo *IDS/IPS*, no intuito de detectar e bloquear tarefas maliciosas.

6.2 Cenário C2 – Uso de Serviços de Bancos de Dados por Aplicações da Grade Computacional

Em outro cenário, deseja-se oferecer o serviço de banco de dados para que tarefas e serviços criem suas bases e/ou utilizem as já existentes. Contudo, uma integração entre ambiente de execução da grade computacional e o serviço de banco de dados apresenta alguns desafios.

Na autenticação, não é conveniente permitir que as aplicações, instanciadas por usuários provenientes de domínios externos, manipulem *logins* e senhas dos servidores locais de banco de dados. Outro problema está na administração: em VOs com grande número de usuários associados dinamicamente, torna-se inviável o gerenciamento de contas e privilégios em todos os bancos de dados envolvidos.

No serviço de autorização, os sistemas de banco de dados possuem modelos complexos e flexíveis, que permitem atribuir privilégios para bancos, tabelas e procedimentos armazenados (*store procedures*). Porém, ao criar um objeto em um banco de dados, o usuário da VO pode querer especificar suas regras de controle de acesso de forma independente da infra-estrutura local (Bell et al., 2002). O GridMultiPolicy pode ser utilizado para a criação de PEPs de granularidade de banco de dados e PDPs relacionados, que implementem modelos de políticas, adicionando assim uma camada de autorização de grade computacional no acesso a tal serviço. A Seção 6.4.5 traz, como exemplo, um modelo de política para banco de dados.

6.3 Cenário C3 – Laboratórios Remotos

Projetos como FusionGrid (FUSIONGRID, 2007), *Earth System Grid* (CHERVENAK et al., 2006) e Neesgrid (KEAHEY et al., 2004) fazem uso da infra-

estrutura da grade computacional para a criação de laboratórios remotos distribuídos. A Figura 44 exemplifica um ambiente de grade computacional de laboratórios remotos, no qual são compartilhados computadores de alto desempenho, armazenamento e distribuição de dados, códigos de aplicações, visualização de experimentos, instrumentos e sensores.

Figura 44 - Ambiente de grade computacional para laboratórios virtuais (FUSIONGRID, 2007).

A utilização dos instrumentos e sensores envolve acesso a recursos locais como arquivos e componentes de *hardware*, podendo ser desenvolvidas aplicações, serviços e bibliotecas específicos para tais tarefas.

Em relação à segurança, o acesso indevido aos recursos pode acarretar prejuízos, tanto em relação ao equipamento que pode ser danificado, como ao comprometimento de dados e informações de outras experiências.

Um ambiente de execução seguro, nesse cenário, deve permitir o uso controlado de recursos como arquivos, dispositivos e sensores. A Figura 45 traz, como solução possível, o uso de GridMultiPolicy para estabelecer políticas no ambiente de execução cuja função é garantir que determinados recursos só sejam acessíveis por meio de bibliotecas específicas, nas quais são configurados os PEPs e PDPs necessários. Para o controle da execução, os PDPs desenvolvidos podem se basear não somente em privilégios associados ao contexto, como em regras para manter a integridade dos dispositivos envolvidos. O estudo de caso apresentado na Seção 6.6 traz um exemplo de política desenvolvida para controlar a realização de experimentos remotos.

Figura 45 – Modelo de controle de acesso para recursos locais específicos.

6.4 Cenário C4 - Controle de Acesso a Serviços de rede da Grade Computacional

Por meio da instancição remota, usuários podem fazer uso da infra-estrutura e de recursos de uma VO para oferecer serviços mais abrangentes. Desta maneira, um determinado usuário pode replicar um serviço em várias instâncias, oferecendo maior disponibilidade, ou mesmo instanciar um serviço em um sítio com o objetivo de oferecer aos outros usuários os recursos ali localizados, de forma mais amigável.

O serviço instanciado por um usuário é uma aplicação que pode ter defeitos de segurança ou mesmo ter sido desenvolvida maliciosamente. Desta maneira, é fundamental exercer um controle de acesso eficiente sobre as operações que o serviço realiza no ambiente de execução. Por outro lado, pode ser necessária a existência de políticas que considerem as características e privilégios do usuário requisitante, para restringir seu acesso no serviço oferecido. A Figura 46 traz uma representação da problemática descrita.

Figura 46 – Controle de acesso em serviços da rede.

Como descrito no capítulo 5, o GridMultiPolicy permite a integração de políticas que tanto podem ter o caráter de controlar as ações realizadas por aplicações no ambiente local de execução, como também as relacionadas ao oferecimento de serviços específicos.

6.5 Implementando Políticas

O estabelecimento de uma política no GridMultiPolicy envolve a realização de quatro tarefas:

1. Desenvolvimento de uma classe Java que implemente a interface PolicyInterface (Figura 28). A classe desenvolvida será utilizada pelo servidor de políticas *MPMS* para gerenciar e distribuir políticas específicas a determinados contextos de execução;
2. Desenvolvimento do PDP correlacionado: classe Java que, além de implementar a interface PDPInterface (Figura 39), deve ser capaz de compreender as políticas no formato adotado para a tomada de decisões de acesso;
3. Caso necessário, especificar instruções para modificações em classes Java no intuito de estabelecer os PEPs necessários ao modelo de política;
4. Configuração no *MPMS*, especificando as classes desenvolvidas nos

dois passos anteriores e adicionando o escopo da política que define o recurso alvo e as ações previstas.

6.5.1 Política LeastPrivilege

A política LeastPrivilege foi realizada como uma abordagem de controle de acesso, visando a efetivação do princípio de privilégio mínimo. Com esse intuito, todas as tarefas e serviços instanciados em sítios da VO são associados a um perfil de segurança, que descrevem os recursos necessários para seu correto funcionamento.

O módulo secundário do MPMS da política LeastPrivilege foi implementado pela classe “br.usp.lsi.grid.PolicyLeastPrivilege”, que manipula arquivos XML para gerenciar suas políticas. No método “CreatePolicy”, a classe “PolicyLeastPrivilege” extrai o parâmetro “profile” do documento RSL e verifica se o usuário possui privilégios para utilizar o perfil selecionado. Em caso positivo, retorna os privilégios correspondentes. O formato de política utilizado foi realizado tendo como base a linguagem XML, prevendo atributos para acesso de arquivos locais, conexões TCP/IP e serviços web. A Figura 47 traz um exemplo de atributos para LeastPrivilege.

```

1  <Action Namespace="http://br.usp.lsi.LeastPrivilege">
2      File_write /tmp/* write</Action>
3  <Action Namespace="http://br.usp.lsi.LeastPrivilege">
4      Socket/lsi.usp.br:7</Action>
5  <Action Namespace="http://br.usp.lsi.LeastPrivilege">
6      WS/lsi.usp.br/axis:research:test</Action>

```

Figura 47 - Exemplo de declaração de privilégios de perfil.

A LeastPrivilege estende a granularidade básica oferecida pelo GridMultiPolicy para, além de controlar as operações em arquivos locais e estabelecer conexões - exercer controle sobre o uso de serviços web. A Figura 48 traz a instrução para a criação de um PEP necessário dentro da biblioteca Axis de serviço web.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <ActionInstruction xmlns="lsi.usp.br/PEPPolicy.xsd"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="lsi.usp.br/PEPPolicy.xsd PEPPolicy.xsd" >
5      <ActionName>webServiceInvoke</ActionName>
6      <PEP>
7          <TypeTarget>Class</TypeTarget>
8          <ClassName>org.apache.axis.client.Call</ClassName>
9          <MethodName>invoke</MethodName>
10         <PropertyName>host:transport.url</PropertyName>
11         <PropertyName>service:operationName.getNamespaceURI()</PropertyName>
12         <PropertyName>port:operationName.getLocalPart()</PropertyName>
13     </PEP>
14 </ActionInstruction>

```

Figura 48 – Instrução para criação do PEP em serviços web na biblioteca Axis .

A classe “br.usp.lsi.LeastPrivilege”, que é uma extensão da interface “br.usp.lsi.PDPInterface”, foi realizada com o PDP capaz de compreender e fazer cumprir as políticas LeastPrivilege.

A realização da leastPrivilege não foi realizada especificamente em nenhum dos cenários discutidos, embora ofereça respostas parciais aos requerimentos de todos. Em relação ao cenário C1, por limitar quais endereços e portas *TCP/IP* estarão disponíveis na aplicação instanciada, pode substituir parcialmente as funcionalidades de mecanismos do tipo filtro de pacotes. Em relação aos cenários C2, C3 e C4, a leastPrivilege pode oferecer um controle de acesso genérico e importante, pois, ao ser complementada por políticas específicas a esses cenários, permite atingir o nível de controle desejado.

6.5.2 Política IDS_Control

Mecanismos do tipo *IDS/IPS* utilizam agentes para detectar e prevenir potenciais vulnerabilidades e ameaças nos níveis de servidor e rede (DEBAR; DACIER; WESPI, 1999). Esses sistemas podem ser implementados em diferentes cenários e com técnicas específicas, como as detecções baseadas em anormalidades ou em assinaturas.

As discussões a respeito de sistemas *IDS/IPS* em ambientes de grade computacional foram abordadas por SCHULTER et al. (2006) e THAM e BUYYA (2005), que identificaram os problemas, os requisitos e as dificuldades da implementação de tais mecanismos de proteção. No entanto, a utilização de *IDS/IPS*

para suporte à decisão de controle de acesso em ambientes de grade não foi encontrada até o momento.

A política IDS_Control (MATTES; MILITELLI; ZUFFO, 2007b) apresentada nesta Seção, tem como objetivo utilizar mecanismos de detecção já implementados no ambiente de rede, de forma a auxiliar o controle sobre as tarefas instanciadas na grade que tenham apresentado comportamento malicioso.

A Figura 49 ilustra a arquitetura proposta para integração, na qual o IDS_Control oferece o serviço Receptor de Detecção de Ataques (RDA) para receber informações de ataque da central de gerenciamento de IDS. Embora não exista um padrão para definição do formato de informações de ataque, o serviço RDA foi elaborado para ser compatível com o formato *IDMEF* (DEBAR; CURRY, 2006), uma vez que este é o único formato que tem sido aceito como um padrão experimental pelo *IETF* (PARK et al., 2003), (MILITELLI, 2006).

A classe “br.usp.lsi.IDSControl” é uma extensão da interface “br.usp.lsi.PDPInterface” e foi concebida como o PDP da política IDS_Control. Todos os acessos à rede que uma tarefa venha a realizar são armazenados em uma tabela de controle. Quando a detecção ocorre, o serviço RDA recebe informações de ataque da central de gerenciamento do IDS, que filtra o endereço *IP* de origem e a porta utilizada, comparando esta informação com a tabela de controle. Uma vez que haja uma correspondência entre as duas informações, é executada uma medida de defesa pré-definida.

A fim de obter uma melhor política de controle de acesso, foram definidos três níveis de medidas de proteção, executadas de acordo com a classificação do ataque: nível baixo - o serviço atacado é bloqueado para aquele determinado endereço de origem; nível médio - toda tarefa é bloqueada para o endereço de origem; e nível alto - o acesso do usuário que submeteu a tarefa ou serviço é negado naquele nó da grade.

Figura 49 – IDS_Control integrado em uma infra-estrutura de IDS/IPS.

O estabelecimento do IDS_control demonstra a capacidade de estabelecer políticas no GridMultyPolicy para promover a integração com mecanismos de segurança já existentes em uma rede. Analisando especificamente o cenário C1, o IDS_Control permite reverter a situação em que o mecanismo IDS/IPS perdia sua efetividade, ao possibilitar seu uso para detectar e bloquear tarefas e serviços que venham a apresentar comportamentos suspeitos.

6.5.3 Política DBModel

Os projetos Spitfire (BELL et al., 2002) e Arquitetura Aberta de Serviços de Grade – Acesso a Dados e Integração (*Open Grid Services Architecture-Data Access and Integration - OGSA-DAI*) (PEREIRA; MUPPAVARAPU; CHUNG, 2006) fazem uso de serviços web para promover a integração entre bancos de dados e a infra-estrutura de grade computacional. Tais soluções, apesar dos avanços oferecidos, aumentam significativamente o tempo gasto na comunicação, além de exigir a reformulação das aplicações que já utilizem banco de dados. Desta maneira, em determinadas situações, pode ser vantajoso permitir que serviços ou tarefas acessem tais serviços sem adicionar-se mais uma camada de comunicação.

A DBModel foi realizada especificamente para o cenário C2, oferecendo um modelo de política que mapeia as conexões das aplicações em *logins* do banco de dados e analisa as requisições realizadas pelas aplicações para adicionar mais uma camada de autorização.

No modelo adotado, no contexto de comunicação com o serviço de banco de

dados, são utilizados perfis de segurança para determinar o identificador (*login*) a ser empregado e os privilégios para acessar objetos. Em tempo de execução, a DBModel reconfigura a *String* de conexão para mapear a aplicação ao *login* escolhido, e posteriormente captura e analisa as requisições SQL para verificar se a aplicação está acessando somente objetos permitidos.

Na plataforma Java, a classe `java.sql.DriverManager` é responsável por gerenciar os *Drivers JDBC* para o estabelecimento de conexões com os bancos de dados. A Figura 50 contém a instrução para a criação do PEP responsável por capturar os parâmetros e repassá-las para que o PDP possa reconfigurar a conexão, mapeando a aplicação em um *login* do banco de dados.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <ActionInstruction xmlns="lsi.usp.br/PEPPolicy.xsd"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="lsi.usp.br/PEPPolicy.xsd PEPPolicy.xsd" >
5      <ActionName>DBConect</ActionName>
6      <PEP>
7          <TypeTarget>Class</TypeTarget>
8          <ClassName>java.sql.DriverManager</ClassName>
9          <MethodName>java.sql.DriverManager</MethodName>
10         <ParameterClass>String</ParameterClass>
11         <ParameterClass>Properties</ParameterClass>
12         <PropertyName>host:$1</PropertyName>
13         <PropertyName>Propriedades:$2</PropertyName>
14     </PEP>
15 </ActionInstruction>

```

Figura 50 – Instrução para a criação do PEP para a ação “DBConect”.

Na plataforma Java, a comunicação com servidores de banco de dados é realizada por classes que implementam a interface “`java.sql.Statement`”, sendo utilizada a linguagem *SQL* para o envio de requisições. A fim de permitir que o DBModel analise todas as requisições *SQL* antes de enviá-las ao banco de dados, foram previstas instruções para criação de PEPs nos métodos “`execute`”, “`executeQuery`” e “`executeUpdate`” em todas as classes que implementam a interface “`java.sql.Statement`” (Figura 51) .

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <ActionInstruction xmlns="lsi.usp.br/PEPPolicy.xsd"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="lsi.usp.br/PEPPolicy.xsd PEPPolicy.xsd" >
5      <ActionName>ExecuteSQL</ActionName>
6      <PEP>
7          <TypeTarget>Interface</TypeTarget>
8          <ClassName>java.sql.Statement</ClassName>
9          <MethodName>executeQuery</MethodName>
10         <PropertyName>SQL:$1</PropertyName>
11     </PEP>
12
13     <PEP>
14         <TypeTarget>Interface</TypeTarget>
15         <ClassName>java.sql.Statement</ClassName>
16         <MethodName>executeUpdate</MethodName>
17         <PropertyName>SQL:$1</PropertyName>
18     </PEP>
19
20     <PEP>
21         <TypeTarget>Interface</TypeTarget>
22         <ClassName>java.sql.Statement</ClassName>
23         <MethodName>execute</MethodName>
24         <PropertyName>SQL:$1</PropertyName>
25     </PEP>
26
27
28  </ActionInstruction>

```

Figura 51 – Instrução para a criação do PEP para a ação “ExecuteSQL”.

O PDP do DBModel foi implementado pela classe Java “br.lsi.usp.DBModel”. A Figura 52 é um exemplo de atributos utilizados pelo DBModel: na primeira linha é definida a permissão para selecionar no “quadro 1”; na segunda, para inserir no “quadro 2”.

```

1  <DBAutorization Action="select"> tabela1</DBAutorization>
2  <DBAutorization Action="insert">tabela2</DBAutorization>

```

Figura 52 – Exemplo de política DBControl.

6.6 Estudo de caso

Com o objetivo de proporcionar um melhor entendimento das capacidades do sistema proposto, esta Seção apresenta um estudo de caso inspirado no projeto NEESGRID. Em face das necessidades de pesquisa em engenharia que envolvem o

uso de equipamentos caros e complexos e a manipulação crescente de volumes de dados, o NEESGRID propõe uma solução em grade para colaboração, em que instituições de pesquisa de engenharia dividem seus equipamentos e o ônus do armazenamento de dados e do processamento. A Figura 53 apresenta, a título de exemplo, fotos de dois equipamentos da Universidade de Illinois utilizados para a realização de experimentos remotos por meio da infra-estrutura de grade computacional.

Figura 53 – Exemplo de equipamentos disponibilizados pela Universidade de Illinois para a realização de experimentos remotos via grade computacional (FUSIONGRID, 2007).

O estudo de caso realizado envolve a instituição fictícia lab.material, que acumulou dados sobre pesquisas de resistência dos materiais e possui um equipamento que permite a realização de experimentos remotos. Ao entrar na VO fictícia “WEBlab”, a lab.Material deseja compartilhar seus recursos com outras instituições, permitindo que as mesmas instanciem tarefas e serviços para fazer uso de equipamentos, dados e aplicações. A

Figura 54 representa os recursos existentes na rede de lab.Material, que é composta pelos seguintes componentes:

- Nós da VO - para que os seus membros, localizados em diferentes regiões do globo, possam instanciar suas tarefas e serviços;
- Serviço de banco de dados - contém dados dos experimentos realizados e informações da corporação lab.Material;
- *Firewall* de rede - oferece serviço de filtro de pacotes como o objetivo de

- proteger a rede corporativa de ataques provenientes da Internet;
- Servidores corporativos - oferecem uma série de recursos que devem ser protegidos;
 - Gerente *IDS/IPS* - coordena uma série de sensores espalhados sobre a rede, no intuito de identificar e bloquear ataques realizados;
 - Instrumentos e sensores - utilizados para a realização de experimentos remotos, devem ter o acesso controlado para evitar uso inadequado.

Figura 54 – Rede corporativa da lab.Material.

Diante da análise realizada sobre os cenários C1, C2, C3 e C4, abordados neste estudo de caso, o grande desafio é permitir que as tarefas e serviços instanciados remotamente possam fazer uso dos recursos de forma controlada, impedindo a realização de danos ao sistema. Nesse sentido, o uso do GridMultiPolicy pode trazer benefícios ao gerenciar e efetivar políticas que aumentem o nível de segurança do ambiente de execução. Como exemplo, podemos citar o benefício da aplicação dos seguintes modelos de políticas:

1. LeastPrivilege, ao permitir que as tarefas e serviços tenham somente acesso aos recursos especificados para seu funcionamento, além de diminuir os eventuais danos que uma aplicação possa vir a realizar, possibilita isolar os recursos que não se deseja compartilhar na VO;
1. IDS_Control, integrando o ambiente de execução da grade

computacional ao sistema *IDS/IPS* existente, é capaz de bloquear tarefas e serviços que apresentem comportamento malicioso ou suspeito;

2. DBControl, oferece uma integração transparente entre o ambiente de execução e bancos de dados, além de adicionar uma camada de autorização com regras da *VO*. Com estas capacidades, permite um controle sobre bancos de dados mais coerente com as necessidades das *VOs*.

Além dos benefícios apresentados, pode-se utilizar o GridMultiPolicy para a efetivação de políticas destinadas a controlar o uso dos sensores e instrumentos da rede lab.Material. Para ilustrar essa possibilidade, foi desenvolvido o modelo de política lab_Control.

6.6.1 Política lab_Control

Esta política foi desenvolvida para exercer controle sobre as experiências remotas que os usuários da *VO* possam realizar dentro da rede lab.Material. O instrumento idealizado teve como inspiração os existentes nos laboratórios do projeto NEESGRID. A Figura 55 representa esse instrumento, que permite aplicar forças em sua estrutura e medir as tensões e ondas mecânicas resultantes. Tal experimento, quando realizado em uma rede colaborativa com diversos instrumentos, permite estudar o comportamento de estruturas complexas, como as existentes em projetos de engenharia civil.

Para a realização do experimento, foi desenvolvida a classe Java “lab.Material.Experimento” que utiliza a biblioteca “Estrutura.experimento.1_0”, supostamente fornecida pelo fabricante, para iniciar e gerenciar a realização de experiências com diferentes configurações.

Figura 55 – Instrumento para execução de testes sobre o comportamento de estruturas.

A política `lab_Control` prevê a criação de PEP na classe “`lab.Material.Experimento`”, garantindo que todas as requisições para a realização de experimentos possam ser analisadas. Deste modo é possível verificar se a aplicação tem autorização para tal realização e se os parâmetros de utilização do dispositivo estão dentro da faixa de segurança aceitável.

No escopo de atuação, as ações da `lab_Control` são relacionadas como de maior granularidade da ação “`permiLoadLib`” da interface “`PDEInterface`” (Figura 28). Foi prevista também uma regra do tipo “`restrict`” para limitar o acesso na biblioteca “`Estrutura.experimento.1_0`” e integrar os dois níveis de granularidade. Dessa maneira, a biblioteca “`Estrutura.experimento.1_0`” será acessível somente pela classe Java “`lab.Material.Experimento`”, que contém os PEPs para a realização da verificação necessária e evita-se, deste modo, que as aplicações utilizem o equipamento sem o controle desejado.

O modelo de política implementado faz uso de perfis de segurança para associar a permissão de realizar experimentos na “`lab.Material`”.

6.6.2 Teste Operacional

Esta Seção apresenta os resultados de um teste operacional inspirado no estudo de caso apresentado, no qual observou-se o comportamento de uma tarefa submetida remotamente, que é controlada pelas políticas desenvolvidas LeastPrivilege, IDS_Control, BDControl e lab_Control.

O ambiente para realização dos testes foi elaborado no sentido de simular a rede como representada pela

Figura 54 e, além do nó da grade computacional, contou com os seguintes componentes:

1. Servidor web - responsável pelo oferecimento dos serviços web na rede de testes, implementado pelo Apache TomCat 6.0.13 (TomCat, 2007).
2. Servidor de banco de dados – foi utilizado o Mysql 5.0 (Mysql, 2007).
3. *MPMS* – Entidade incumbida de gerenciar as políticas do GridMultiPolicy;
4. Mecanismo do *tipo* IDS -- foi considerado o uso do *Snort* (SNORT, 2006) como mecanismo de análise e detecção de ataques e da extensão *Snort-IDMEF* (POPPI, 2007), para permitir que o *Snort* enviasse mensagens de informação de ataques, de acordo com o formato *IDMEF* ao serviço *RDA* da política IDS_Control.

O Quadro 5 relaciona os privilégios assegurados aos perfis de segurança adotados pelos modelos de políticas no teste realizado.

Modelo de Política	Regra
LeastPrivilege	Ler - File://temp/*
	Acessar: - TCP/IP: 192.168.0.77:22
	Acessar – serviço web : http://lsi.usp.br/axis teste
BDControl	Selecionar - Tabela: Tabela 1
IDS_control	Não aplicavel
Lab_control	Executar Experimento – carga < 1000

Quadro 5 - Privilégios utilizados para realização de testes.

A fim de verificar a capacidade de efetivação das políticas propostas, destinada a permitir a correta execução de tarefas remotas, sem comprometimento da segurança de rede lab.Material, foi concebida uma aplicação para testes. Esta, submetida remotamente no nó da grade, foi programada para realizar duas seqüências de ações, com intervalo de 1 minuto entre elas. A última ação da primeira seqüência simula um ataque ao servidor web.

Para a submissão da aplicação remota, foi criado o arquivo descritor de tarefas “test_Lab.rsl” (Figura 56), no qual o valor “test_PDE” foi atribuído ao parâmetro “*profile*”, que define o perfil de segurança a ser utilizado. Desta forma, ao utilizar o programa “globusrun-ws” do GT4 para instanciar as ações de acordo com o arquivo de tarefa “test_PDE.rsl”, a aplicação submetida começa a usufruir dos privilégios deste perfil.

```

1  <job>
2    <executable>java</executable>
3    <argument>org.atacante.PesquisaMaliciosa</argument>
4    <profile>test_PDE</profile>
5    <required_resource>DataBase_service</required_resource>
6    <required_resource>Virtual_lab</required_resource>
7    <fileStageIn>
8      <transfer><sourceUrl>
9        gsiftp://192.168.0.44:2811/home/user/teste.jar
10       </sourceUrl>
11       <destinationUrl>
12         file:///${USER_HOME}/teste.jar
13       </destinationUrl>
14     </transfer>
15   </fileStageIn>
16 </job>

```

Figura 56 – Descrição do arquivo de tarefa “test_PDP.rsl”

Conforme mostra a

Figura 57, durante a execução da primeira seqüência, foram permitidas somente as ações previstas para o “test_PDP”, de acordo com o PDP “LeastPrivilege”. Uma vez que o Snort identificou o ataque ao servidor web, transmitiu esta informação ao RDA que, por sua vez, classificou o ataque como nível médio. Com base nesta classificação, o PDE “IDS_Control” passa a bloquear todas as operações da tarefa.

6.7 Conclusões

O estudo realizado sobre cenários de uso de uma grade computacional ponderou as vulnerabilidades e as necessidades de segurança. Em seguida, por meio do GridMultiPolicy, foram desenvolvidas políticas que permitem aumentar o nível de segurança nos cenários estudados.

O estudo de caso apresentado foi inspirado no projeto NEESGRID, que utiliza a grade computacional para o compartilhamento de recursos e a realização de experimentos remotos. Nesse contexto, o uso de GridMultiPolicy confere maior flexibilidade e segurança ao sistema, permitindo o estabelecimento de regras específicas para controlar os diferentes tipos de recursos envolvidos.

O teste de funcionalidade simulou o comportamento de uma aplicação, no ambiente simulado, que, além de tentar realizar uma série de ações não permitidas, executou um ataque a um servidor da rede. Os resultados demonstraram a capacidade do GridMultiPolicy para fazer cumprir as múltiplas políticas por ele gerenciadas, impedindo ações não permitidas e bloqueando a aplicação após a realização do ataque.

Os cenários de uso, o estudo de caso e o teste de funcionalidade foram realizados a fim de proporcionar uma visão prática do uso do GridMultiPolicy que, dada a sua flexibilidade, permite desenvolver e efetivar políticas com diferentes finalidades, granularidades e formatos de regras. Embora não tenha sido possível explorar todas as possibilidades de uso do GridMultiPolicy, foi porém comprovada a sua capacidade de oferecer resposta a diferentes requisitos de segurança presentes numa grade computacional.

```

[testeusu@curupira ~]$ globusrun-ws -submit -s -f test_PDE.rsl
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:2a5ctty2e-c19b-11dr-a93
getting the Axis.jar
getting the LeastPrivilege and DBcontrol policy
Current job state: Active
Current job state: CleanUp-Hold

***** primeira série !*****
Escrevendo em: /tmp/GridCliente
Falha ao escrever em : /bin/malicioso
Conexão realizada em : 192.168.0.77:7 !
Falha ao conectar-se em 192.168.0.22:1024
Selecionado dados da tabela 1.
Falha ao selecionar dados da tabela 2.
Experiência executada com carga = 30
Falha em executar Experiência com carga = 100
Falha ao tentar acessar o serviço web: test em http://lsi.usp.br/axis
acessando serviço web: admin em http://lsi.usp.br/axis

***** Segunda série !*****
Falha ao escrever em : /tmp/GridCliente
Falha ao escrever em : /bin/malicioso
Falha ao conectar-se em : 192.168.0.77:7 !
Falha ao conectar-se em 192.168.0.22:1024
Falha ao selecionar dados da tabela 1.
Falha ao selecionar dados da tabela 2.
Falha em executar Experiência com carga = 30.
Falha em executar Experiência com carga = 100.
Falha ao tentar acessar o serviço web: teste em http://lsi.usp.br/axis.
Falha ao tentar acessar o serviço web: admin em http://lsi.usp.br/axis.

*****

Current job state: Done || Destroying job...Done.

```

Figura 57 – Resultado da execução da tarefa “test_PDE.rsl”.

7 Análise de desempenho

Já se sabe que oferecer segurança requer recursos computacionais. Ao acrescentar um mecanismo para proteger um sistema computacional é de se esperar um decréscimo em seu desempenho. A viabilidade de um mecanismo envolve a correlação entre seus benefícios e seu custo associado. Para estimar o real benefício que um mecanismo agrega, deve-se ponderar sobre o nível de proteção por ele oferecido e a relevância dos recursos envolvidos. Já na estimativa do custo associado, além dos fatores relativos à sua implementação, devem-se contabilizar os recursos computacionais por ele consumidos. Deste modo, a viabilidade de um mecanismo ou plataforma não é um conceito absoluto e depende das características do ambiente no qual se planeja adotá-lo (ABRAMS; JAJODIA; PODELL, 1995). Portanto, apesar da análise de desempenho de determinado mecanismo não validá-lo de forma absoluta, ela permite estabelecer condições em que seu uso é proveitoso.

A análise apresentada neste capítulo tem como objetivo oferecer dados para estabelecer as condições em que se recomenda o uso do GridMultiPolicy. Neste sentido, foram realizados testes de desempenho, simulando a utilização de uma grade computacional em diferentes condições de uso.

A estimativa do desempenho do sistema proposto é um desafio, dada a dificuldade de prever as possíveis peculiaridades que tarefas, serviços e políticas possam assumir. Os testes realizados limitaram-se a estudar aplicações que simulam a realização de operações básicas, quando controladas pelos modelos de políticas apresentados no Capítulo 6.

O GridMultiPolicy é um sistema de controle de acesso, sendo seu uso possível somente com a utilização conjunta de outras tecnologias que ofereçam uma infra-estrutura de grades computacionais. Os testes foram realizados com as integrações descritas nas Seções 5.4.1 e 5.4.2, nas quais o GridMultiPolicy é empregado para exercer controle de acesso nos serviços de grade computacional *WS Gram* e *Java WS Core*.

7.1 Ambiente de Testes

Os testes foram realizados em estrutura de rede representada pela Figura 58, que conta com servidores web e de banco de dados, uma estação cliente, um nó da VO, o gerenciador de soluções de segurança *IDS/IPS* e o servidor de políticas do GridMultiPolicy *MPMS*. As máquinas representadas foram interligadas por meio de um *Switch FastEthernet*.

No Quadro 6 temos a configuração das máquinas utilizadas quanto a hardware, sistema operacional, serviço oferecido e componentes de *software* adicionais.

Figura 58 – Rede estabelecida para a realização de testes.

A fim de avaliar o impacto da utilização do GridMultiPolicy no desempenho das aplicações de uma grade computacional, foram realizados estudos e testes sobre essa topologia durante a instanciação e execução de tarefas e no oferecimento de serviços.

Entidade	Equipamento	Sistema Operacional	Serviço Oferecido	Componentes de Software Instalados
Cliente	<i>Pentium</i> 4 1.6 GHz 512 Mb	Linux 2.6	Nenhum	Aplicações clientes para utilização dos serviços do <i>GT4</i>
Servidor Web	<i>Pentium</i> 4 1.6 GHz 512 Mb	Linux 2.6	<i>http</i>	Apache Tomcat 6.0
Nó da grade	<i>Pentium</i> 4 1.6 GHz 512 Mb	Linux 2.6	Instanciação remota de tarefas e hospedagem de serviços web	Serviços <i>WS GRAM</i> e <i>Java WS core</i> do <i>GT4</i> integrados ao GridMultiPolicy
Gerente <i>IDS</i>	<i>Pentium</i> 4 1.6 GHz 512 Mb	Linux 2.6	Detecção e prevenção de ataques	<i>Snort</i> 2.6 com o modulo <i>Snort-IDMEF</i>
<i>MPMS</i>	<i>Pentium</i> 4 1.6 GHz 512 Mb	Linux 2.6	Distribuição de políticas em múltiplos formatos	Serviço <i>MPMS</i> do GridMultiPolicy
Servidor de banco de dados	<i>Pentium</i> 4 1.6 GHz 512 Mb	Linux 2.6	Serviço de banco de dados	<i>Mysql</i> 5.0

Quadro 6 - Equipamentos utilizados nos testes práticos.

7.2 Avaliação do Impacto de Desempenho na Instanciação e Execução de Tarefas Remotas

Utilizando a rede teste, montada conforme a topologia descrita no item anterior, foram realizados testes para medir o impacto sobre a execução remota de tarefas na grade computacional. O desempenho da utilização do GridMultiPolicy

depende diretamente dos tipos de políticas envolvidas, que variam conforme as necessidades de uma *VO*, os sítios e aplicações. Os testes relativos à instanciação de tarefas foram realizados utilizando políticas *LeastPrivilege*, *IDS_Control* e *DBMODEL*, descritas na Seção 6.5.

Conforme a Seção 5.4.1, o processo de execução de tarefas remotamente envolve quatro etapas:

1. Processamento da requisição;
2. Configuração do ambiente de execução;
3. Iniciação;
4. Execução das operações.

Desta maneira, o tempo total associado à realização de uma tarefa remotamente é a soma dos tempos gastos nas quatro etapas relacionadas.

No primeiro teste realizado, foi utilizado o *WS Gram* integrado ao *GridMultiPolicy* e mediu-se o tempo para o estabelecimento do ambiente de execução de tarefas. Essa etapa engloba a autenticação e o processamento da requisição de submissão de tarefa, a aquisição das políticas pertinentes ao contexto dado e a criação do ambiente de execução configurado. A Figura 59 traz o gráfico comparativo do tempo gasto na instanciação de tarefas entre a utilização do *GridMultiPolicy* com diferentes tamanhos de políticas e do ambiente *Grade* em sua forma convencional, sendo realizado 500 repetições para cada configuração estudada. Pode observar uma sobrecarga entre 212 e 254 *mili* segundos.

Figura 59 – Tempo médio de instanciação remota de tarefas utilizando diferentes tamanhos de políticas.

Outros testes foram realizados no sentido de medir o impacto da utilização do GridMultiPolicy na realização de operações pelas tarefas instanciadas. A Figura 60 traz o gráfico comparando o tempo de execução de operações de escrita de mensagens em arquivos locais, quando utilizadas políticas com diferentes números de regras de acesso, para cada configuração estudada foram realizadas 500 repetições e medidas. Observa-se, nesse ensaio, uma sobrecarga de 21% a 33%, diretamente relacionada ao número de regras das políticas utilizadas. Além disso, quanto maior o tamanho da mensagem a ser escrita, menor a sobrecarga proporcional.

Figura 60 – Tempo para escrita em arquivos locais com e sem a utilização do GridMultiPolicy.

Nos testes para mensurar a sobrecarga no uso de conexões *TCP/IP* foram utilizadas mensagens de 10 bytes a 10 Kbytes e políticas com diferentes números de regras. A Figura 61 mostra os resultados médios de 1000 operações de rede e apresenta a relação entre o tempo gasto utilizando e não utilizando o GridMultiPolicy. No pior caso, quando foram enviadas mensagens pequenas utilizando políticas com 100 regras, observa-se sobrecarga inferior a 20 %.

Figura 61 – Sobrecarga proporcional na utilização de conexões *TCP/IP*.

Também foram realizados testes focados em modelos de políticas específicas. Instanciando remotamente uma tarefa que simulava um ataque ao servidor web, enviando aos servidores uma mensagem que é cadastrada como maliciosa no *IDS/IPS*. Mediu-se o tempo gasto no processo de detecção pelo *IDS/IPS* e do envio do alarme para o mecanismo RDA da política *IDS_Control*. O resultado desse teste é demonstrado no Quadro 7. Pode-se concluir que, quando uma tarefa ou serviço da grade realiza uma ação considerada maliciosa, o sistema age com relativa rapidez, restando pouco tempo para a realização de novos ataques, antes de a tarefa ser bloqueada.

Tempo de 100 detecções	Média por detecção
64856 ms	648 ms

Quadro 7 – Tempo para detecção de ataques.

A Figura 62 traz o gráfico resultante das medidas do impacto da política DBModel no acesso a banco de dados. Na execução dos testes, foram utilizadas políticas com diferentes números de regras, tendo sido medido o tempo de acesso a diferentes números de registros de uma tabela exemplo, constituída por um campo único do tipo “*Varchar(250)*” e possuindo 10000 registros, sendo realizado medidas sobre 500 repetições de acesso. Os resultados mostram que, no pior caso, utilizando

políticas com 100 regras e acessando somente 10 registros, houve uma sobrecarga relativa inferior a 10% .

Figura 62 – Sobrecarga proporcional no acesso a banco de dados.

O serviço de banco de dados, utilizado por boa parte das aplicações, é o fator de gargalo em muitas delas. Os testes realizados demonstram a capacidade do GridMultiPolicy em efetivar políticas de acesso a banco de dados em grades computacionais com sobrecarga relativamente baixa.

7.3 Avaliação do Impacto na Instanciação e Oferecimento de Serviços

Os testes para mensurar o impacto no oferecimento de serviços de rede computacional foram realizados utilizando o ambiente de hospedagem de serviços *Java WS Core*, integrado ao GridMultiPolicy, descrito na Seção 5.4.2.

No *Java WS Core*, o ambiente de execução de um contêiner é iniciado uma única vez e utilizado por múltiplos usuários, podendo ocorrer a criação de um contexto de segurança para cada usuário que requisite um serviço, caso haja a necessidade. Dessa maneira, os testes de desempenho foram focados na iniciação de um serviço e no gerenciamento de contextos.

Para a realização dos testes, além dos modelos de políticas LeastPrivilege, IDS_Control e DBModel, utilizou-se uma nova política, desenvolvida especificamente para simular diversas configurações no controle de acesso em

serviços web.

Na instanciação do serviço do contêiner, as políticas pertinentes ao contexto são obtidas junto ao servidor *MPMS* para, em seguida, ser estabelecido o ambiente de execução adaptado. A Figura 63 traz o gráfico com os tempos de estabelecimento do ambiente de execução, utilizando políticas com diferentes tamanhos, no qual se observa uma sobrecarga de 210 a 274 ms.

Figura 63 – Tempo para iniciação do contêiner no *Java WS Core*.

O *Java WS Core* oferece um serviço web com informações de estado (*stateful*), no qual o contexto de um usuário é gerado na sua primeira conexão e armazenado para ser utilizado no processamento das próximas requisições. Conforme descrito na Seção 5.2.3, a interface *PDPInterface* (Figura 24) pode ser utilizada para que os PDPs das políticas façam uso dos contextos gerados pelo *Java WS Core* para gerenciar seus próprios contextos, podendo assim controlar o acesso a serviços, conforme as características e os privilégios dos usuários.

Na realização de um contexto, o PDP, caso necessário, pode requerer novas políticas com regras pertinentes às características do usuário requisitante. A Figura 64 traz os resultados das medidas de tempo para estabelecimento de contexto em três situações:

- Sem uso do sistema de segurança;
- Utilizando o *GridMultiPolicy* com PDP que não necessite de novas políticas para a criação de contexto;

- Utilizando o GridMultiPolicy com um PDP requisitando políticas com diferentes tamanhos para a realização de contexto.

Figura 64 – Tempo para criação de contextos.

Os testes demonstraram que, sem a requisição de novas políticas, o desempenho não sofre grandes alterações. No entanto, quando novas políticas se fazem necessárias, observa-se uma sobrecarga entre 213ms e 316 ms. Como o estabelecimento de contextos ocorre somente na primeira requisição de um usuário, a sobrecarga não afeta o oferecimento do serviço como um todo e as requisições subseqüentes não sofrem esse encargo.

7.4 Conclusões

Este capítulo apresentou um estudo sobre o impacto no desempenho que a utilização do GridMultiPolicy em ambiente de grade computacional pode representar. Para a realização dos testes que serviram de base para a análise, utilizaram-se os serviços de grade computacional *WS Gram* e *Java WS Core* integrados ao GridMultiPolicy para gerenciamento e efetivação das políticas de segurança desenvolvidas no capítulo 6.

A análise dos testes realizados aponta que a utilização do GridMultiPolicy impacta de forma não desprezível a instanciação, iniciação e execução de tarefas e serviços da grade computacional. Na instanciação de serviços e tarefas, o impacto de desempenho se deve à obtenção de políticas e configuração do ambiente de

execução. Já os testes para medir a sobrecarga nas ações realizadas por tarefas e grades controladas pelo GridMultiPolicy indicam que as operações de acesso a conexões *TCP/IP* e banco de dados apresentaram uma sobrecarga proporcional inferior a 20%, e as de acesso a arquivos locais, entre 20% e 33%. A significância dos resultados obtidos depende das características dos sistemas envolvidos. Contudo, pode-se afirmar que o GridMultiPolicy com as políticas apresentadas impõe um impacto relativamente pequeno para a maior parte das aplicações, exceto em aplicações que necessitem de alto desempenho e tenham acessos a arquivos locais, como fator de gargalo.

A realização de uma avaliação de desempenho definitiva do sistema proposto não pode ser realizada, dada a dificuldade de prever as características de uso de uma grade computacional e os possíveis modelos de políticas que possam ser adotados. Os testes de desempenho apresentados neste capítulo tiveram como base as políticas descritas no capítulo 6, que foram desenvolvidas para oferecer respostas a problemas e requisitos presentes em cenários de uso de uma grade computacional. Desta maneira, apesar das limitações apresentadas, os testes estão diretamente relacionados com as condições reais de uso, e seus resultados fornecem dados importantes para estimar o impacto que o GridMultiPolicy e suas políticas possam exercer sobre as aplicações em uma *VO*.

8 Conclusões, Limitações e Futuros Trabalhos

Este capítulo contém as conclusões obtidas durante a elaboração desta tese, as limitações ainda existentes nas soluções propostas e os direcionamentos que motivarão futuros trabalhos de continuidade ao desenvolvimento do GridMultiPolicy.

8.1 Conclusões

O tema “grade computacional” surgiu há mais de dez anos, com a perspectiva de se tornar a tecnologia computacional do futuro, base para sistemas mais flexíveis, robustos, integrados e colaborativos. Desde então, apesar de avanços notáveis, tal previsão ainda não se realizou e as grades computacionais são utilizadas no momento quase que exclusivamente na área acadêmica. A ausência de mecanismos de segurança apropriados pode ser indicada como um dos fatores que impedem a disseminação maciça dessas tecnologias. A adoção de um modelo de maior integração e colaboração exige capacidades não encontradas nos sistemas de segurança vigentes.

Esta tese descreveu o sistema GridMultiPolicy com o intuito de oferecer um controle de acesso coerente com as necessidades das grades computacionais, promovendo o gerenciamento, a efetivação e a integração de múltiplas políticas, cada qual com seu formato, granularidade, mecanismo e objetivos. O GridMultiPolicy destaca-se também por permitir a realização conjunta de políticas para controlar tanto o acesso a serviços de rede, como as ações requisitadas por aplicações (tarefas ou serviços) no ambiente de execução.

A utilização de técnicas de programação conhecidas, algumas específicas da plataforma Java, permitiu atingir muitas das capacidades do GridMultiPolicy:

- O uso de escopos de atuação permite especificar o contexto de uso das políticas em uma VO. Desta maneira, um sítio ou um usuário, ao se associarem a uma VO, podem definir os mecanismos e as políticas que serão utilizados para proteger os recursos por eles disponibilizados;
- A especificação de interfaces de comunicação capacitou o adcionamento de novos componentes ao GridMultiPolicy, de modo a gerenciar e efetivar novas políticas com seus formatos, linguagens e mecanismos envolvidos;

- O emprego de técnicas para manipulação de classes Java em tempo de execução permite, sem alterar o código-fonte das aplicações, estabelecer ambientes de execução configurados, com os pontos de checagem necessários às granularidades das políticas previstas.

O GridMultiPolicy foi concebido para ser utilizado de forma integrada com diferentes plataformas de grade computacional. Dessa forma, não especifica mecanismos para estabelecer e gerenciar relações de confiança em uma VO, deixando a cargo da plataforma prover tal funcionalidade. Esta característica foi evidenciada na integração realizada, tendo sido utilizado o *GT4* para prover os serviços básicos de uma grade computacional, que inclui o estabelecimento de relações de confiança, e o GridMultiPolicy para gerenciar e efetivar as políticas de controle de acesso.

A análise de cenários de uso aprofundou a discussão a respeito das vulnerabilidades e necessidades de segurança das grades computacionais, servindo de base para a realização de políticas que demonstraram a capacidade do GridMultiPolicy para oferecer respostas às necessidades de segurança de uma VO. Podem-se destacar as funcionalidades das seguintes políticas desenvolvidas:

1. A “leastPrivilege” possibilita a efetivação do princípio de privilégio mínimo, oferecendo um controle mais conciso sobre as aplicações, sem interferir em seu correto funcionamento;
2. A IDS_Control demonstra a capacidade de interligar o ambiente de execução da grade computacional com mecanismos existentes, no caso, utilizando um *IDS/IPS* para detectar e bloquear tarefas e serviços da grade que apresentem comportamento malicioso;
3. A DBModel permite que as aplicações da grade computacional possam acessar, de forma controlada, bancos de dados locais em sítios de uma VO;
4. A labControl é um exemplo de como o GridMultiPolicy pode ser utilizado para controlar a utilização de equipamentos em laboratórios remotos.

As avaliações práticas funcionais e de desempenho realizadas são limitadas, diante da impossibilidade de prever todos os possíveis modelos de políticas necessários em uma grade computacional. Contudo, são de grande importância para estimar as capacidades e o custo computacional associado ao uso do GridMultiPolicy. Os resultados de desempenho apontam para a viabilidade do sistema, uma vez que o acréscimo menor que um segundo no tempo de

estabelecimento do ambiente de execução de serviços e tarefas e uma sobrecarga proporcional menor que 20%, medidos na maior parte das operações, podem muito bem ser suportados pela maior parte das aplicações. Como destaque negativo, as operações de acesso a arquivos locais apresentaram uma sobrecarga proporcional entre 20% e 34%. Estes valores, que podem não ser de grande significância na maior parte das situações, podem, contudo, inviabilizar o uso do GridMultiPolicy em aplicações que exigem alto desempenho e tenham um fator de gargalo nesse tipo de operação.

Concluindo, esta tese apresentou um panorama de segurança nas grades computacionais, apontando algumas de suas principais vulnerabilidades. O sistema proposto apresentou como principal contribuição um modelo de controle de acesso para o gerenciamento e efetivação de políticas distintas. As análises realizadas deflagraram a capacidade do GridMultiPolicy para atingir seus objetivos, em conjunto com as políticas desenvolvidas e seus mecanismos relacionados, promovendo respostas adequadas a muitas das necessidades de segurança das grades computacionais ainda não supridas.

8.2 Limitações

Durante a concepção do sistema proposto nesta tese e sua correspondente implementação, algumas limitações foram identificadas. Apresentamos as principais a seguir.

A primeira foi a escolha da tecnologia Java para a realização do ambiente de execução *JMPE*. Se, por um lado, essa tecnologia se coaduna com os preceitos da grade computacional de heterogeneidade, permitindo a sua utilização em diferentes sistemas operacionais, por outro, é uma limitação, criando incompatibilidade com aplicações não-Java.

Os testes funcionais a respeito do uso da grade computacional para a realização de experiências remotas foram realizados sobre equipamentos fictícios. A idéia inicial era a utilização dos ambientes e as aplicações do projeto correlacionado NEESGRID, oferecendo assim um experimento sobre uma condição real de uso. Contudo, tal realização não foi possível, primeiro, porque a documentação sobre o projeto ainda é escassa e, em segundo lugar, por falta de disponibilidade das instituições envolvidas. Após alguns contatos iniciais, os pesquisadores do

laboratório da Universidade de Illinois se mostraram favoráveis a permitir a utilização da infra-estrutura disponível. No entanto, devido à prioridade de alguns projetos em andamento naquela Universidade, não puderam se comprometer em fornecer ajuda efetiva.

8.3 Futuros Trabalhos

Esta seção apresenta algumas direções que podem originar futuros trabalhos, aumentando a abrangência e as capacidades até aqui apresentadas.

Como mencionado na seção anterior, uma das limitações do presente trabalho é não oferecer suporte a aplicações não-Java. Outras pesquisas poderiam estudar a viabilidade da implementação do modelo de segurança proposto, utilizando outros ambientes de execução. Um caminho tecnológico possível para tal realização é a utilização do Módulo Linux de Segurança (*Linux Security Module*) (SMALLEY; VANCE; SALAMON, 2007), um conjunto de interfaces de programação que permitem alterar o setor do núcleo do sistema operacional Linux que realiza as decisões de controle de acesso.

As políticas de segurança apresentadas ofereceram respostas a necessidades presentes nas grades computacionais, contudo, estão longe de esgotar o assunto. A flexibilidade do sistema proposto, além de contribuir para a segurança das grades computacionais, abre possibilidades de estudar seu uso com diferentes modelos de políticas e mecanismos. Trabalhos futuros que realizem políticas com maior eficácia e/ou capacidades, podem trazer contribuições importantes.

REFERÊNCIAS BIBLIOGRÁFICAS

.NET: Plataforma .NET. Disponível em: <<http://msdn2.microsoft.com/pt-br/netframework/default.aspx>>. Acesso em: 06 jan. 2007.

APACHE TOMCAT. Disponível em : <<http://tomcat.apache.org/>>. Acesso em: 06 jan. 2007.

ADAMS, C.; S. Farrell. **RFC2510 – Internet X.509 Public Key Infrastructure Certificate Management Protocols**, 1999. Disponível em: <<http://www.cis.ohiostate.edu/htbin/rfc/rfc2510.html>>. Acesso em: 06 jan. 2007.

AXIS: Biblioteca de Web Service. Disponível em : <<http://ws.apache.org/axis/>>. Acesso em: 06 jan. 2007.

ABRAMS, D; JAJODIA, S.; PODELL, J. **Information security** : an integrated collection of essays. Los Aamitos, Calif. : IEEE Computer Society Press, c1995.

BAVIER A.; BOWMAN, M. **Operating System Support for Planetary-Scale Network Services**, NSDI'04, San Francisco, US, 2004.

BRIAN S.; GRIMSHAW, S.; NGUYEN-TUONG, A. **Grid-Based File Access: The Legion I/O Model**, In: PROCEEDINGS OF THE NINTH IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, Pittsburgh, USA, pp. 165-174, August 2000.

CHERVENAK, A.; et al. **Monitoring the Earth System Grid with MDS4**. In: 2th IEEE INTERNATIONAL CONFERENCE ON E-SCIENCE AND GRID COMPUTING (e-Science 2006), Amsterdam, 2006.

CHADWICK, D.; OTENKO, O. **The PERMIS X.509 Role Based Privilege Management Infrastructure**. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES (SACMAT 2002), Monterey, 2002.

COETZEE, M.; ELOFF, J. **Virtual enterprise access control requirement**. In: SOUTH AFRICAN INSTITUTE OF COMPUTER SCIENTISTS AND INFORMATION TECHNOLOGISTS (SAICSIT 2003), Indaba, 2003.

CORBASEC: Corba Security Specification. Disponível em: <www.corba.com>. Acesso em: 06 jan. 2007.

DAMIANI, E.; VIMERCATI, S.; SAMARATI, P. **New paradigms for access control in open environments**. In: The 5th IEEE INTERNATIONAL SYMPOSIUM ON SIGNAL PROCESSING AND INFORMATION, Athens, Greece, 2005.

DEBAR, H.; CURRY, D.; FEINSTEIN, B. **RFC4765: The Intrusion Detection Message Exchange Format**. (IDMEF), IETF, 2006.

DEBAR, H.; DACIER, M.; WESPI, A. **Towards a taxonomy of intrusion-detection systems**. J. Computer Networking, v. 31, n. 9, p. 805-822, 1999.

DODONOV, E; QUAINI, J.; GUARDIÃ, H. **Gridbox: Securing Hosts from Malicious and Greedy Applications**. In: MIDDLEWARE FOR GRID COMPUTING, Toronto, 2004.

FOSTER, I. **Globus Toolkit Version 4: Software for Service-Oriented Systems**. In: IFIP - INTERNATIONAL CONFERENCE ON NETWORK AND PARALLEL COMPUTING, SPRINGER-Verlag LNCS 3779, p. 2-13, 2005.

FUSIONGRID: Projeto para integração de pesquisas científicas. Disponível em: <<http://www.fusuongrid.org>>. Acesso em: 06 jan. 2007.

GRIDPMA: Autoridade internacional de gerenciamento de políticas. Disponível em: <<http://gridpma.org>>. Acesso em: 06 jan. 2007.

GOLDCHLEGER, A.; KON, F.; GOLDMAN, A.; FINGER, M.; BEZERRA, G. C. . **InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines**. Concurrency and Computation-Practice & Experience, EUA, v. 16, p. 449-459, 2004.

HERZOG, A.; SHAHMEHRI, A. **Using the Java Sandbox for Resource Control**. In: 7th NORDIC WORKSHOP ON SECURE IT SYSTEMS. Karlstad, Sweden, p. 135-147, 2005.

HUMPHREY, M.; THOMPSON, M.; JACKSON, K. **Security for Grids**. In: IEEE, v. 93, n. 3, p. 644-652, 2005.

JAJODIA, S.; SUBRAHMANIAN, V. **A Logical Language for Expressing Authorizations**. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY (S&P'97), 1997.

JAVA: Plataforma Java. Disponível em: <<http://java.sun.com>>. Acesso em: 06 jan. 2007.

JAVA WS CORE – Serviço de hospedagem de aplicações de serviços web. Disponível em: <<http://www.globus.org/toolkit/docs/4.0/common/javawscore/>>. Acesso em: 06 jan. 2007.

JAVASSIST: Biblioteca para manipulação de Java ByteCode. Disponível em <<http://www.csg.is.titech.ac.jp/~chiba/javassist/>>. Acesso em: 06 jan. 2007.

JAX-WS: Biblioteca de *Web Service*. Disponível em: <<http://java.sun.com/webservices/jaxws/index.jsp>>. Acesso em: 06 jan. 2007.

KEAHEY, K.; et al. **Grids for Experimental Science: The Virtual Control Room**. In: PROCEEDINGS OF THE SECOND INTERNATIONAL WORKSHOP ON CHALLENGES OF LARGE APPLICATIONS IN DISTRIBUTED ENVIRONMENTS, Hawaii, 2004.

KEAHEY, K.; RIPEANU, M.; DOERING, K. **Dynamic Creation and Management of Runtime Environments in the Grid**. In: WORKSHOP ON DESIGNING AND BUILDING WEB SERVICES (GGF 9), Chicago, 2004.

LOCKHART, Jr. **OSF DCE Guide to Developing Distributed Applications**. New York: McGraw-Hill, 1994.

LORCH, M.; KAFURA, D. **The PRIMA Grid Authorization System**. Journal of Grid Computing, v. 2, n. 3, 2004.

MATTES, L.; ZUFFO, J. **Access control platform for submitted jobs in communication** In: NETWORK, AND INFORMATION SECURITY (CNIS 2006), Cambridge, 2006.

MATTES, L.; et al. **Linguagem lógica formal para expressar segurança em ambientes pervasivos**. In: IV WORKSHOP EM SEGURANÇA DE SISTEMAS COMPUTACIONAIS, Porto Alegre, 2004.

MATTES, L.; MILITELLI, L.; ZUFFO, J. **Plataforma para efetivação de múltiplas políticas de controle de acesso em ambientes de grade computacional**. A ser publicado em: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS (SBSeg), Rio de Janeiro, p.27-31, agosto de 2007.

MATTES, L.; MILITELLI, L.; ZUFFO, J. **Platform to enforce multiple access control policy in grid hosting environment**. A ser publicado em: FIRST INTERNATIONAL WORKSHOP ON SECURITY, TRUST AND PRIVACY IN GRID SYSTEMS (Grid-STP 2007), Nice, France, September 17, 2007.

MARSHALL D. A.; JAJODIA, S.; PODELL H. P. **Information security** : an integrated collection of essays. Los Aamitos, Calif. : IEEE Computer Society Press, c1995.

McNAB, A. **Grid-Based access control and user management for Unix environments, file systems, web sites and virtual organizations**. In: CHEP 2003, La Jolla, Calif., 2003.

MILITELLI, L. C. **Proposta de um agente de aplicação para detecção, prevenção e contenção de ataques em ambientes computacionais**. Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

MEINEL C. A. **Framework for supporting distributed access control policies**. In: 10th IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC'05), p. 442-447, 2005.

MySQL: Servidor de banco de dados. Disponível em: < <http://www.mysql.com/> >. Acesso em: 06 jan. 2007.

NAGARATNAM, N.; et al. **The security architecture for open grid services**, Version 1. July 17, 2002.

NABRZYSKI ,L.; SCHOPF, J. **Grid resource management** : state of the art and future trends, Kluwer Publishing, 2003.

NOVOTNY, J.; TUEKE, S. **An online credential repository for the grid** : MyProxy. In: INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING (HPDC-10), 10., 2001.: proceedings. IEEE Press, 2001.

OTENKO, S.; CHADWICK, D. A. **Comparison of the akenti and permis authorization infrastructures**, 2003. Disponível em:
<<http://sec.isi.salford.ac.uk/download/AkentiPERMISDeskComparison2-1.pdf>>.
Acesso em: 06 jan. 2007.

PARK, S.; HUMPHREY, M. **Authorizing remote job execution based on job properties**. In: 2th IEEE INTERNATIONAL CONFERENCE ON E-SCIENCE AND GRID COMPUTING (e-Science 2006), Amsterdam, 2006.

PARK, S.; KIM, K.; JANG, J.; NOH, B. **Supporting interoperability to heterogeneous IDS in secure networking framework**. In: THE 9TH ASIA-PACIFIC CONFERENCE ON COMMUNICATIONS, v. 2, p. 844-848, 2003.

PEARLMAN, L.; WELCH, V.; FOSTER, I.; KESSELMAN, C. **The community authorization service: Status and Future**. In: COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS (CHEP03), La Jolla, USA, 2003.

PEREIRA, A.; MUPPAVARAPU, V.; CHUNG, M. **Role-Based access control for grid database services using the community authorization service**. In: IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, v. 3, n. 2, 2006.

PINHEIRO JÚNIOR, J. R.; KON, F. **Representando opiniões em cadeias de confiança SPKI/SDSI**. In: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS (SBSeg), Santos, Setembro, 2006.

PINHEIRO JÚNIOR, J. R.; VIDAL, A. T.; KON, F. **Repositório seguro de aplicações baseado em GSS**. In: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS (SBSeg), Florianópolis, Setembro, 2005.

PISTOIA, M.; RELLER, F.; GUPTA, D. **Java 2 network security**. Prentice-Hall, 2nd Edition, 2005.

POPPI, S. **Snort-IDMEF plug-in**. Disponível em:
<<http://sourceforge.net/projects/snort-idmef>>. Acesso em: 06 jan. 2007.

SAML: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) Version 2.0. Disponível em: <<http://www.oasis-open.org/committees/>>.
Acesso em: jan. 2007.

SANDHU, R.S.; SAMARATI, P. "Access Control: Principles and Practice", IEEE Computer, p. 40-48, Sept 1994. Disponível em:
<<http://citeseer.ist.psu.edu/article/sandhu94access.html>>. Acesso em: 06 jan. 2007.

SCHNEIER, B. **Applied cryptography**: protocols, algorithms and source code in C. 2. ed. New York: John Wiley & Sons; 1996.

SCHULTER, A.; REIS, J. A.; KOCH, F.; WESTPHALL, C. B. **A Grid-Based intrusion detection system**. In: INTERNATIONAL CONFERENCE ON SYSTEMS AND INTERNATIONAL CONFERENCE ON MOBILE COMMUNICATIONS AND LEARNING TECHNOLOGIES (ICNICONSMCL'06), 2006.

STALLINGS, W. **Cryptography and network security** : principles and practice. 2. ed. Prentice Hall, 1998. 570 p.

STEENBAKKERS, M. **Guide to LCAS**, Version 1.1.16. Setembro, 2003.

STEVE, L.; CARLISLE, A. **Understanding PKI**: concepts, standards, and deployment considerations. Boston. Addison-Wesley, 2002.

SNORT- Ferramenta de IDS. Disponível em : <<http://www.snort.org/>>. Acesso em: 06 jan. 2007.

SMALLEY, S.; VANCE, C.; SALAMON, W. **Implementing SELinux as a Linux security module**. Disponível em:
<<http://www.nsa.gov/selinux/papers/module/t1.html>>. Acesso em: jan. 2007.

THAM, C. K.; BUYYA, R. **SensorGrid** : Integrating Sensor Networks and Grid Computing. Special Issue on Grid Computing, Computer Society of India, 2005.

VOLLBRECHT, J. et al. "AAA Framework", Internet RFC2904, Internet Engineering Task Force, Network Working Group, August 2000.

W3C. Disponível em: <<http://www.w3.org/>>. Acesso em: 06 jan. 2007.

KERBEROS Protocolo de autenticação. Disponível em:
<<http://web.mit.edu/Kerberos/>>. Acesso em: 06 jan. 2007.

WANG, Y. **SPKI**. 1998. Disponível em:
<<http://www.hut.fi/~yuwang/publications/SPKI/SPKI.html>>. Acesso em: jan. 2007.

WEDDE, H. F.; LISCHK, A. M. **Modular authorization**, In: ACM SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES (SACMAT2000), 2000., proceedings. ACM, 2000.

WELCH, V.; et al. **Security for Grid Services**. In: Twelfth INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING (HPDC-12), IEEE Press, 2003.

WELCH, V.; et al. **Attributes, Anonymity, and Access: Shibboleth and Globus integration to facilitate grid collaboration**. Proceedings of the 4th Annual PKI R&D Workshop, 2005.

XACML: Extensible Access Control Markup Language (XACML) Version 2.0. Disponível em: <<http://www.oasis-open.org/committees/xacml>>. Acesso em: jan. 2007.