Plataforma para efetivação de múltiplas políticas de controle de acesso em ambientes de grade computacional

Leonardo Mattes, Leonardo C. Militelli, João Antonio Zuffo

Laboratório de sistemas integráveis – Universidade de São Paulo (USP)

Caixa Postal 15.064 – 91.501-970 – São Paulo – SP – Brazil

{leo, leonardo, jazuffo}@lsi.usp.br

Abstract. *A coherent access control service for Grid environment must combine multiple policies allowing administrators, sites and users to determine specific rules to protect its resources. This paper proposes a flexible access control framework based on Java for Grid environment that permits management, effectuation and integration of multiple policies and existent security mechanisms. In order to show this framework's capabilities it was deployed an operational test to manage and put into effect two policies: one based on the least privilege principle and a second that integrates access control service to an IDS security system.*

Resumo. Um serviço de controle de acesso em ambientes de Grades computacionais coerente deve combinar múltiplas políticas, permitindo com que administradores, sítios e usuários determinem regras e mecanismos para proteger seus recursos. Esse trabalho propõe uma plataforma de controle de acesso flexível para aplicações Java em ambientes de grade computacional, que permite o gerenciamento, efetivação e integração de múltiplas políticas e mecanismos de segurança. No intuito de demonstrar as capacidades do sistema proposto foi realizado um teste operacional para gerenciar e efetivar de forma integrada duas políticas: uma que integra o controle de acesso a um sistema IDS local e outra focada no princípio de privilegio mínimo.

1 Introdução

O termo grade computacional faz referência a uma classe de sistemas distribuídos, que permite a associação e integração de múltiplos domínios independentes organizações virtuais (VOs) [Foster, 2005], [Bavier et al 2004]. Um serviço de controle de acesso eficiente e integrado sobre uma plataforma heterogênea é um desafio de grande importância para as grades computacionais. Seu correto oferecimento deve contribuir para o uso de um modelo de maior integração, sem agravar os riscos dos sistemas envolvidos [Welch et al, 2003].

Um típico ambiente de grade computacional oferece um ambiente unificado e integrado sobre múltiplos domínios, no qual os recursos são oferecidos e gerenciados dinamicamente, sem a intervenção direta de administradores. No entanto, deve permitir que cada domínio ou usuário desenvolva e gerencie seus próprios mecanismos e políticas de forma autônoma [Foster, 2005], [Nabrzyski e Schopf, 2003]. Desta maneira, um serviço de autorização para grades computacionais coerente deve oferecer suporte a múltiplas políticas, que exerça um controle unificado sobre os recursos disponibilizados

e ofereça autonomia para que domínios e usuários estabeleçam os mecanismos e políticas para controlar seus recursos.

Para demonstrar a importância dos requisitos descritos, considere a análise dos cenários a seguir de forma conjunta. No primeiro, o administrador de uma VO deseja implementar um sistema de controle de acesso que utiliza perfis de segurança para a efetivação do princípio do privilégio mínimo [Saltzer e Schroeder, 1975]. No segundo, em um sítio de uma VO, um determinado administrador deseja integrar um mecanismo IDS com o ambiente de execução, a fim de detectar e bloquear tarefas maliciosas. Tal solução teria grande valor para diminuir os riscos associados ao serviço da grade computacional de instanciação de tarefas, que pode representar uma grande vulnerabilidade aos sistemas [Keahey, Ripeanu e Doering, 2004].

Esse artigo apresenta GridMultiPolicy, plataforma[1] de controle de acesso flexível para aplicações Java em ambientes de grade computacional, que permite o gerenciamento, efetivação e integração de múltiplas políticas e mecanismos de segurança. No modelo proposto, políticas independentes baseadas em suas próprias regras, mecanismos e níveis de granularidade são integradas por meio de interfaces de programação bem definidas e de acordo com um escopo de atuação. Para a efetivação das múltiplas políticas o sistema estabelece dinamicamente um ambiente customizado, de acordo com o contexto de execução.

No intuito de demonstrar como a plataforma proposta pode ser utilizada para atingir os objetivos de segurança da grade computacional, este artigo apresenta a integração de duas políticas que: 1) oferece resposta ao princípio do privilégio mínimo, um dos requisitos de segurança em grade computacional, segundo Welch [2003]; 2) integra o ambiente de execução com um mecanismo IDS, aumentando a segurança do serviço de instanciação remota de tarefas, por meio do uso da infra-estrutura já estabelecida em uma rede hipotética.

2 Trabalhos relacionados

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 produzidos 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 Shibboleth [Welch et al, 2005], Permis [Otenko e Chadwick, 2003] e Gridmap [Gridpmap, 2005] 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 integrar políticas locais de um sítio ao processo de autorização, utilizando uma interface de comunicação. Estes trabalhos têm em comum uma limitação. Suas efetivações exigem o uso de aplicações confiáveis e desenvolvidas de forma específica a cada plataforma. Desta maneira, podem ser muito bem utilizadas nos serviços inerentes a grade computacional, oferecidos por aplicações

[1] O termo plataforma é utilizado no sentido de oferecer uma base tecnológica para o estabelecimento de múltiplas políticas de segurança.

próprias. Contudo, a sua utilização em ambientes de execução de tarefas não confiáveis, não foi apresentada.

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

- Os trabalhos [Lorch e Kafura, 2004] e [Burruss et al, 2006] adotam modelos com políticas complexas para a criação e gerenciamento de contas Unix locais, utilizadas na execução das tarefas. Por mais flexíveis que as políticas adotadas sejam, sua efetivação é realizada por mecanismos locais, nos quais os recursos disponíveis podem variar de acordo com a grade computacional utilizada.
- Outras pesquisas [Park e Humphrey, 2006] e [Mattes e Zuffo, 2006] fazem uso de *sandboxes* (caixas de areia) 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.

A plataforma proposta, comparada com os trabalhos estudados, contribui para a segurança de grades computacionais em dois aspectos: oferece um serviço de controle de acesso flexível com suporte ao gerenciamento, efetivação e integração de múltiplas políticas e permite seu uso tanto no oferecimento de serviços, como em ambiente de execução de tarefas.

3 Desafios para integração de múltiplas políticas

Pesquisas recentes [Damiani, Vimercati e Samarati, 2005] [Coetzee e Eloff, 2003] apresentam estudos sobre a efetivação de múltiplas políticas de segurança independentes, apontando os seguintes desafios para a sua realização:

- Oferecer suporte ao gerenciamento e utilização de regras representadas em diferentes formatos ou linguagens;
- Permitir que cada política especifique seus próprios mecanismos de efetivação;
- Estabelecer os controles necessários para a efetivação da granularidade especificada por cada política. A granularidade fina é um conceito relativo que varia de contexto para contexto. Em uma determinada política ou situação, se faz necessário controlar as operações de ler e escrever arquivos do sistema, já em outra, controlar os acessos a objetos dentro de um arquivo XML [Damiani, Vimercati e Samarati, 2005].

Outro fator importante a ser considerado 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 existir políticas com alvos e granularidades distintas que contemplam recursos computacionais em comum.

Tradicionalmente, políticas de granularidade distintas são efetivadas de forma independente, o que nem sempre atende aos requisitos necessários. A figura 1 ilustra tal problemática, em que, no sistema de segurança representado coexistem duas políticas de controle de acesso com diferentes níveis de granularidades. A primeira prevê as operações tradicionais de ler, modificar e apagar arquivos do sistema; a segunda controla as operações de acesso ao conteúdo dos objetos de documentos XML. Para

assegurar que determinados arquivos sejam corretamente protegidos, as duas políticas devem trabalhar de forma integrada, impedindo que determinados arquivos se tornem acessíveis sem o controle da política para XML.

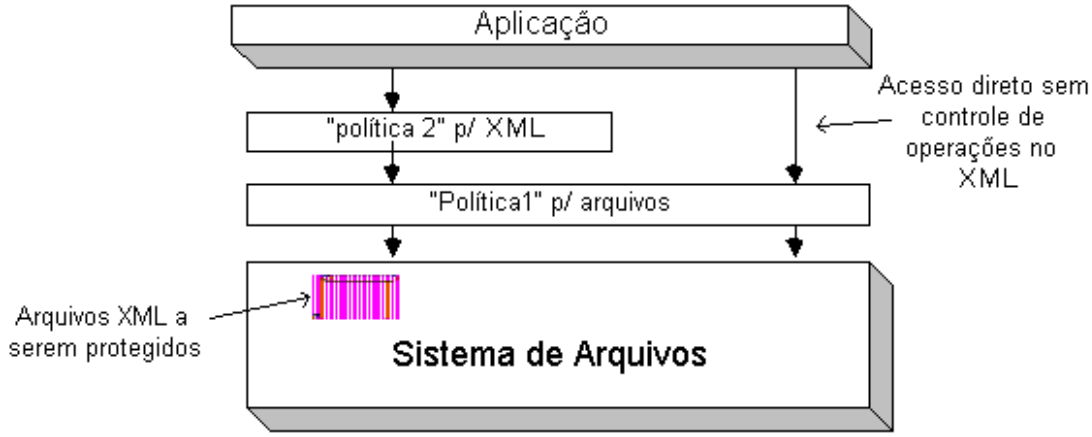

Figura 1. Problemática da integração de diferentes níveis de granularidade.

4 GridMultiPolicy

O GridmultiPolicy propõem um modelo de controle de acesso em que administradores e usuários podem estabelecer suas políticas de forma independente para cada alvo a ser protegido. O alvo, que é o objetivo de cada política, pode ser um serviço, um sítio, uma VO ou um determinado recurso. No modelo proposto, as tarefas ou serviços são inicializados em um ambiente de execução customizado, capaz de fazer cumprir as políticas relevantes ao dado contexto.

O processo de autorização utilizado teve como base a especificação AAA [Vollbrecht et al, 2000], no qual são previstos PEPs (*Policy Enforcement Point* -Pontos de Cumprimento de Política), PDP (*Policy Decision Point*- Ponto de Decisão de Política) 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. Desta maneira, o processo de autorização proposto envolve o gerenciamento dos PEPs e PDPs envolvidos.

A figura 2 contém a arquitetura Gridmultipolicy com seus dois componentes integrados, o MPMS (*Multi Policy Manager Service* – Serviço de Gerenciamento de Múltiplas Políticas), responsável por gerenciar políticas de segurança em múltiplos formatos e o JMPE (*Java Multi Policy Environment* – Ambiente Java de Múltiplas Políticas), incumbido de oferecer um ambiente de execução customizado com os PEPs e PDPs necessários para fazer cumprir as políticas estabelecidas para um dado contexto.

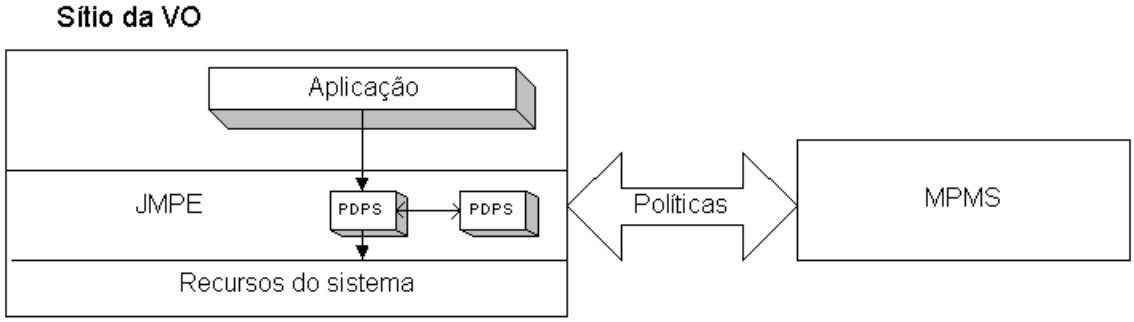

Figura 2. Arquitetura GridMultiPolicy

4.1 MPMS

A figura 3 representa a estrutura do MPMS, que conta com um módulo principal para o gerenciamento das regras gerais de controle de acesso e múltiplos módulos secundários. Cada módulo secundário fornece suporte a políticas em um determinado formato ou modelo, estando associado a um escopo para delimitar sua atuação, definindo as ações previstas e o recurso alvo.

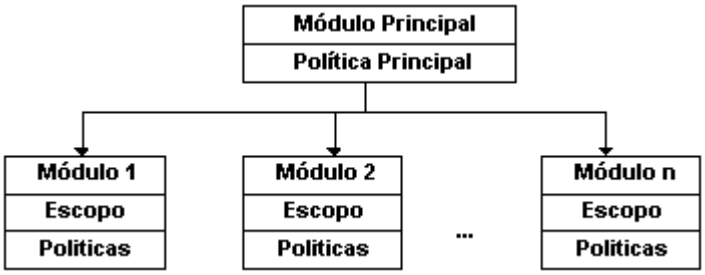

Figura 3. Estrutura hierárquica do MPMS

Para a representação das políticas do módulo principal e dos escopos de atuação, foi utilizada a linguagem XACML [XACML, 2007] (padrão OASIS) com duas pequenas modificações, que permitiram oferecer suporte as regras para a integração de ações de granularidade hierarquicamente correlacionadas. Na primeira alteração, adicionou-se a capacidade de representar relacionamento entre ações correlacionadas. Na segunda, acrescentou-se o tipo de regra "Restrict", utilizado para condicionar o acesso a determinados recursos ao uso de uma política com um nível de granularidade maior.

No exemplo fornecido na seção 3 (integração da política XML com a de sistema de arquivos), as ações XML deveriam ser hierarquicamente relacionadas com as de acesso a arquivos do sistema, de granularidade menor. Para garantir o correto uso da política XML, os arquivos a serem protegidos devem ser controlados, na política de granularidade menor, por regras do tipo "Restric".

O serviço MPMS, para distribuição de múltiplas políticas, recebe como parâmetro um arquivo RSL (*Resource Specification Language* - Linguagem para Especificação de Contexto) descritor de tarefa e retorna um certificado X.509, contendo políticas em múltiplos formatos. O RSL é um padrão da grade computacional para descrição de tarefas, prevendo atributos como nome do programa a executar, serviços a serem utilizados, bibliotecas necessárias, parâmetros de iniciação, etc. No processo de geração do certificado de múltiplas políticas, o MPMS, com base no arquivo RSL,

analisa quais módulos possuem escopos relevantes ao dado contexto, e por meio da interface "PolicyInterface" da figura 4, submete o arquivo RSL aos módulos secundários, para receber como resposta as políticas específicas nos formatos utilizados. Em seguida, o MPMS realiza a credencial de acesso contendo as políticas obtidas.

```
public interface PolicyInterface
    public byte[] createPolicy(RSL Description, VOUser user);
```

Figura 4. Interface "PolicyInterface" para a comunicação entre o módulo principal com os secundários, na geração das credenciais de acesso específicas aos contextos.

4.2 JMPE

O JMPE é a entidade responsável pelo estabelecimento de um ambiente de execução customizado, com os PEPs e PDPs necessários para oferecer suporte as múltiplas políticas previstas.

As definições, políticas e as classes Java de todos os PDPs previstos são obtidas junto ao MPMS durante a iniciação do JMPE em um sítio., Posteriormente, como mostra a figura 5, antes de executar uma tarefa ou serviço, com base no arquivo RSL associado, o JMPE estabelece um ambiente de execução customizado com os PDPs pertinentes ao contexto.

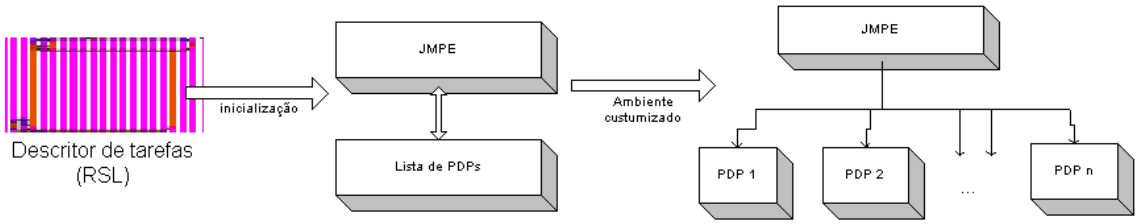

Figura 5. Processo de criação dos PDPS de um ambiente de execução.

Os PEPs são pontos de checagem, posicionados entre a aplicação e os recursos, definidos para oferecer suporte a um nível de granularidade. O JMPE contém PEPs pré-estabelecidos e permite o estabelecimento dinâmico de outros, de modo a oferecer suporte a diversos níveis de granularidade.

Por padrão, as aplicações Java são controladas somente pelo sistema operacional. No entanto, existem meios de atribuir a uma aplicação um gerente de segurança capaz de controlar as ações por meio de uma série de pontos de checagem existentes na plataforma Java [Pistoia et al, 1999]. O JMPE faz uso de um extensão do gerente de segurança Java para obtenção dos PEPs de granularidade básica, que os tornam habilitados a exercer controle sobre as ações de acesso a arquivos locais e estabelecimento de conexões TCP/IP.

Para o estabelecimento de PEPs de uma determinada granularidades se faz necessário atuar dentro de seu contexto. 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 à banco de dados etc. Desta maneira, o desafio para o estabelecimento de controle a um nível de granularidade é a criação dos PEPs necessários dentro de um dado contexto. O JMPE permite associar a cada ação prevista, instruções para modificar bibliotecas em tempo de execução, no intuito de estabelecer os PEPs necessários a uma dada granularidade.

A figura 6 traz como exemplo a instrução para o estabelecimento de um PEP para uma granularidade de *Web Service* que, por meio de modificações na biblioteca Axis [AXIS, 2007], permite inserir um ponto de checagem para autorizar toda a chamada ao método "invoke" da classe "org.apache.axis.client.Call".

```
<PEP action="webServiceInvoke" grain="webService" classtarget="org.apache.axis.client.Call" >
   <Method="org.apache.axis.client.Call   <Method name="invoke">
      <property name="host" value="transport.url"/>
      <property name="service" value="operationName.getNamespaceURI()"/>
      <property name="port" value="perationName.getLocalPart()"/>
</Method></PEP>
```

Figura 6. Instrução para criação do PEP na biblioteca Axis.

Para a realização das instruções de modificações de bibliotecas, o JMPE possui o JMPECLassLoader, entidade responsável em carregar e manipular classes Java em tempo de execução. A figura 7 descreve o processo em que as classes são carregadas e modificadas dinamicamente, por intermédio dos seguintes passos:

1. Na fase de estabelecimento do ambiente de execução, com base nas ações descritas nos escopos das políticas, se estabelece uma lista contendo instruções para modificação de classes;
2. Em tempo de execução, a JVM (*Java Virtual Machine* - Máquina Virtual Java) envia uma requisição para carregar uma determinada classe;
3. O JMPECLassLoader carrega do sistema de arquivos o Java *bytecode* associado ao nome da classe;
4. Verifica a existência de instruções associadas à classe carregada;
5. Caso necessário, utiliza a biblioteca Javassist [JavaAssist, 2007] para manipular os *Bytecodes* e estabelecer o PEP previsto;
6. Retorna à JVM os *Bytecodes* da classe, possivelmente alterados.

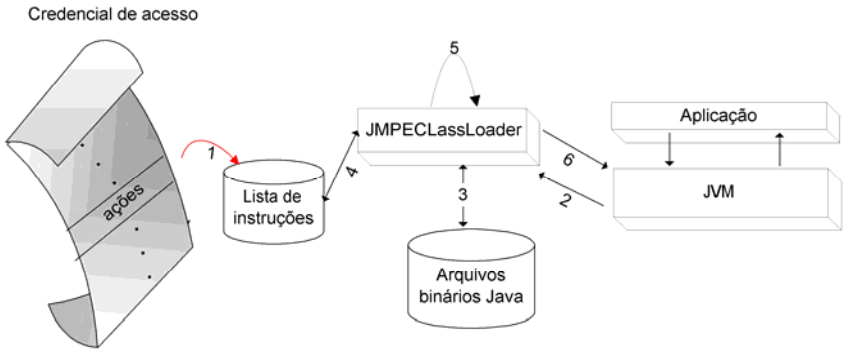

Figura 7. Processo de carregamento de classes no JMPE.

4.3 Processo de autorização

No processo de autorização, os PEPs consultam os PDPs para negar ou permitir as ações, por intermédio do JMPE. A comunicação entre o JMPE e os PDPs é realizada pela interface da "br.usp.lsi.PDPInterface" (figura 8), que deve ser implementada por todos os PDPs. Os métodos "permitFile" e "permitSock" são utilizados, respectivamente, para verificar a autorização de acesso aos arquivos locais e

estabelecimento de conexões TCP/IP. O método "permitOperation" é genérico, sendo utilizado para autorizações de ações de outras granularidades.

```
public interface PDPInterface {
        public PDPAutorization permitFile(String filename,  int operation, Object executionContext) ;
        public PDPAutorization permitSock(String host, int port,     Object context) ;
        public PDPAutorization permitOperation(String operation, Object[] parameters) ; }
```

Figura 8. Interface PDPInterface

A cada requisição de autorização recebida o JMPE prossegue por meio dos seguintes passos:

1. Coleta as decisões dos PDPs pertinentes ao contexto;
2. Em caso de conflito das decisões, aplica o algoritmo de resolução de conflitos;
3. Autoriza ou nega a ação.

Para a resolução de conflitos são previstos quatros algorit*mos "deny-overrides", "allow-overrides", "first-applicable" e "only-one-applicable"*, especificados pelo trabalho XACML 2007.

Caso o PDP retorne *"Restrict"*, como resposta a uma tomada de decisão, estará definindo que o recurso deve ser permitido somente com emprego de uma política de granularidade maior. Baseando-se nos relacionamentos entre as ações e na lista de classes Java modificadas, o JMPE verifica se a requisição da ação é proveniente de uma classe com PEP de granularidade maior. Caso positivo, a ação é autorizada, ao contrario, é negada.

Utilizando mais uma vez o exemplo fornecido na seção 3, os arquivos a serem controlados pela política de XML e protegidos por regras "restrict", serão acessíveis somente por classes modificadas com os PEPs de granularidade XML.

5 Teste operacional

No intuito de oferecer uma visão prática das capacidades da plataforma de controle de acesso proposta, esta seção apresenta um teste operacional realizado na plataforma GT4 integrada ao GridMultiPolicy, envolvendo duas políticas de segurança desenvolvidas.

5.1 Integrando GridMultiPolicy ao GT4

A plataforma GridMultiPolicy foi concebida para ser utilizada de forma integrada a diversos sistemas e infra-estruturas. Esta seção mostra como foi realizado a integração do GT4 (*Globus Tool Kit*, versão 4.2) [Foster, 2005] ao GridMultiPolicy para a execução do teste operacional.

O GT4 consiste em um conjunto de ferramentas e serviços relacionados, cuja integração oferece uma infra-estrutura para grades computacionais. O serviço web para locação e gerenciamento de recursos WS GRAM (*Web Service Grid Resource Allocation and Management*) do GT4, permite que usuários de uma VO instanciem remotamente tarefas. A figura 9 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 descrevendo a tarefa e seu certificado X.509, para autenticação;

2. O WS GRAM processa transferências dos arquivos necessários às tarefas e/ou dos mecanismos para a efetivação das políticas;
3. O WS-GRAM envia para JMPE uma requisição de execução da tarefa;
4. Caso seja necessário, novas políticas são obtidas no MPMS e a tarefa é iniciada;
5. Após a finalização da tarefa, outras transferências de arquivos podem ocorrer;

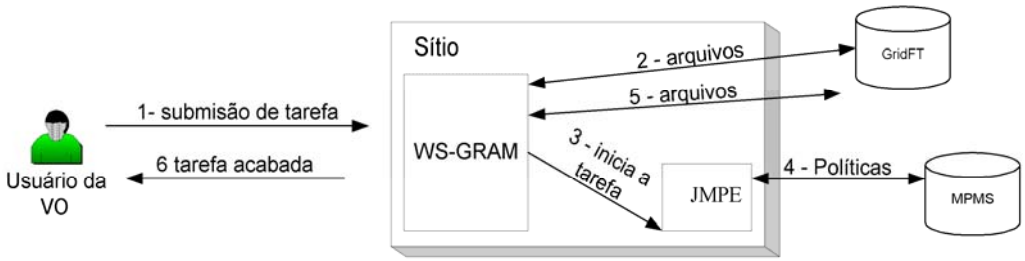

Figura 9. Processo de submissão de tarefas

5.2 Implementando Políticas

Para adicionar uma política no GridMultiPolicy, são necessárias três etapas:

1. Desenvolvimento de uma classe Java que defina a interface "PolicyInterface" e gerencie as políticas no formato adotado;
2. Desenvolvimento do PDP correlacionado, classe Java que implementa a interface "PDPInterface" e seja capaz de compreender e fazer cumprir as políticas estabelecidas;
3. Configuração da política no MPMS, especificando as classes desenvolvidas nos dois passos anteriores e adicionando um arquivo XACML como escopo da política. O escopo fornecido deve definir o recurso alvo, as ações e as instruções para o estabelecimento dos PEPs necessários.

- ***Política LeastPrivilegeModel***

O objetivo do modelo de política LeastPrivilegeModel é a efetivação do princípio do privilégio mínimo. Com esse intuito, toda tarefa da VO é associada a um perfil de segurança que define os mínimos recursos necessários para seu correto funcionamento.

No modelo adotado, os certificados dos usuários contêm privilégios para a utilização de perfis. Na fase de submissão de tarefas, o usuário define no arquivo RLS a intenção de utilizar um determinado perfil.

O módulo secundário para o gerenciamento da política do privilégio mínimo foi implementado pela classe "br.usp.lsi.grid.PolicyLeastPrivilege", que faz uso de arquivos escritos segundo padrão SAML [SAML 2007]. Foram previstos atributos para autorizar ações sobre arquivos locais, conexões TCP/IP e *Web Service*.

O controle de acesso ao sistema de arquivos local e estabelecimento de conexões TCP/IP já é previsto pela granularidade básica dos PEPs utilizados pelo JMPE. Contudo, se faz necessário a criação de PEPs para ações específicas do *Web Services.* A figura 8, da seção 4.2, traz as instruções para a criação do PEP de *Web Service* na biblioteca AXIS. Com o uso de regras "Restrict", pode-se restringir o acesso de determinados endereços TCP/IP ao uso das bibliotecas modificadas com suporte a granularidade *Web Service*.

- ***Política IDS_Control***

Sistemas de detecção e prevenção de intrusos (IDS/IPS) utilizam agentes para detectar e prevenir possíveis vulnerabilidades e ameaças no nível de servidor e rede [Debar, Dacier e Wespi, 1999]. Estes sistemas podem ser implementados em diferentes cenários e com técnicas específicas de detecção, como as baseadas em anomalidades ou assinaturas.

Os trabalhos [Schulter et al, 2006], [Tham e Buyya, 2005] trazem uma discussão a respeito de sistemas IDS em ambientes de grade computacional, identificando os problemas, os requisitos e as dificuldades da implementação de tais mecanismos. No entanto, a utilização de IDS para suporte à decisão de controle de acesso em ambientes de grade não foi encontrada até o momento.

A política IDS_Control, 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 10 apresenta a proposta da arquitetura de integração, na qual o IDS_Control oferece o serviço GER (*Grid Event Receiver*- Receptor de eventos da grade) para receber informações de ataque da central de gerenciamento de IDS. Embora não exista um padrão para descrição de informações de ataque, o serviço GER foi elaborado para ser compatível com o IDMEF [Debar, Curry e Feinstein, 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", como o PDP da política IDS_Control. Todos os acessos à rede que uma tarefa instanciada tenha realizado são armazenados em uma tabela de controle. Quando a detecção ocorre, o serviço GER recebe informação 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, todas as ações da tarefa são bloqueadas.

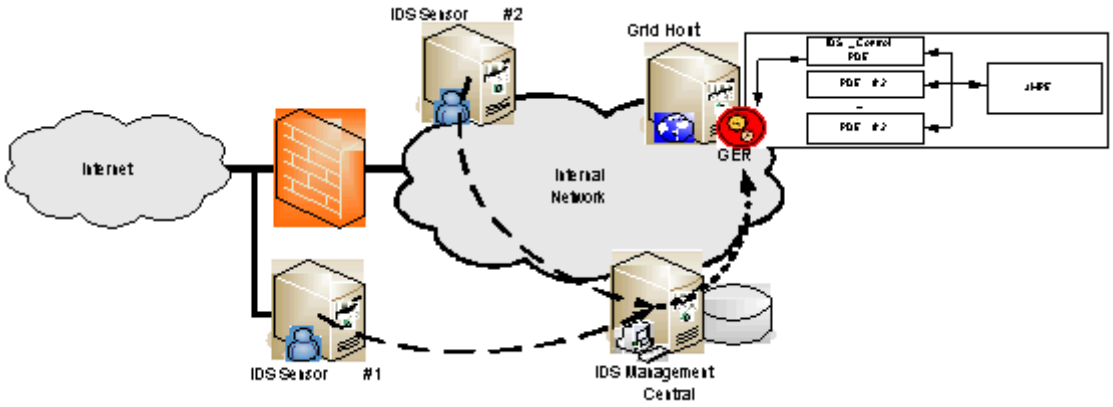

Figura 10. IDS_Control integrado em uma infra-estrutura de IDS/IPS.

5.3 Executando o teste

O teste operacional foi realizado por uma tarefa Java submetida remotamente, programada para a execução de operações. Para o contexto da submissão realizada, especificou-se o uso das políticas LeastPrivilegeModel e IDS_Control.

A figura 11 contém os atributos definidos pelo perfil "teste" da política "LeastPrivilege", permitindo operações de leitura e escrita no diretório "/tmp", conexões TCP e UDP para o servidor 192.168.0.77 na porta 7 e acesso ao *Web Service* "Test" em qualquer porta, a partir da URL "http://lsi.usp.br/axis".

```
<Action Namespace="http://br.lsi.usp.br/LeastPrivilege"> File_write /tmp/* write</Action>
<Action Namespace="http://br.lsi.usp.br/LeastPrivilegeModel">TCP 192.168.0.77:7</Action>
<Action Namespace="http://br.lsi.usp.br/LeastPrivilegeModel">WS lsi.usp.br/axis:teste:*</Action>
```

Figura 11. Arquivo de políticas do perfil "teste"

Na submissão da aplicação, utilizou-se o arquivo de tarefa "teste.rsl" (Figura 12), que especifica o uso do perfil "teste" atribuído pelo parâmetro "project".

```
<job>
     <executable>java</executable>
    <argument>br.usp.lsi.TestePermissao</argument>
     <project>test_PDE </project>
      <fileStageIn><transfer><sourceUrl>gsiftp://192.168.0.44:2811/home/user/testeGlobus.jar</sourceUrl>
      <destinationUrl>file:///${GLOBUS_USER_HOME}/testepermissao.jar</destinationUrl></transfer>
</fileStageIn>
</job>
```

Figura 12. Descrição do arquivo de tarefa "teste.rsl"

Para analisar e detectar ataques no teste realizado, utilizou-se o IDS Snort [Snort, 2006], em conjunto com seu módulo Snort-IDMEF [Poppi, 2006]. O uso desta extensão permitiu o envio de informações sobre as detecções realizadas ao GER, segundo o formato IDMEF.

A aplicação "br.usp.lsi.TestePermissao" submetida, foi programada para realizar duas seqüências de ações. A última ação da primeira seqüência simula um ataque a um servidor web da rede.

[testeusu@curupira ~]$ globusrun-ws -submit -s -f test_PDE.rsl Delegating user credentials...Done. Submitting job...Done. Job ID: uuid:2a5ctyy2e-c19b-11dr-a93 … getting the LeastPrivilegePDP.jar getting the IDSPDP.jar getting the Axis.jar getting the LeastPrivilege policy Current job state: Active Current job state: CleanUp-Hold ********** first round !********** Write in /tmp/GridCliente fail to write /bin/malicioso	Connected to 192.168.0.77:7 ! fail to Connect to 192.168.0.22:1024 . fail WS "admin" in http://lsi.usp.br/axis WS " test" in http://lsi.usp.br/axis ********** second round !********* fail to write /tmp/GridCliente2 fail to write /bin/malicioso2 fail to conect to 192.168.0.77:7 fail to conect to 192.168.0.22:1024 . fail WS "admin" in http://lsi.usp.br/axis fail to WS "test" in http://lsi.usp.br/axis End of the test! ********** Current job state: Done || Destroying job...Done.

Figura 13. Resultado da execução da tarefa "teste.rsl"

Conforme a Figura 13, durante a execução da primeira seqüência foram permitidas somente as ações previstas pelo perfil “teste”, de acordo com o LeastPrivilegeModel. Uma vez que o Snort identificou o ataque ao servidor web e transmitiu a informação ao GER, a política IDS_Control passou a bloquear todas as operações da tarefa, conforme o planejado.

6 Considerações finais

O oferecimento de segurança em grades computacionais exige o estabelecimento de um ambiente integrado sobre múltiplos domínios independentes, mesclando políticas globais com locais. Como resposta a estes requisitos, este trabalho apresentou GridMultiPolicy, plataforma de controle de acesso para grades computacionais que permite o gerenciamento, efetivação e integração de múltiplas políticas.

O desenvolvimento da plataforma proposta foi realizado com base em tecnologias Java específicas. O uso de interfaces de comunicação possibilita incorporar suporte a novos formatos de políticas e mecanismos ao GridMultiPolicy. A utilização de instruções para manipulação de classes Java em tempo de execução, permite que, sem alterar o código fonte das aplicações ou bibliotecas, se possa estabelecer dinamicamente diferentes níveis de granularidade.

O teste operacional desenvolvido teve como objetivo, demonstrar como o GridmultiPolicy pode trabalhar de forma integrada a uma plataforma de grade computacional e oferecer uma primeira visão de suas capacidades, mediante o desenvolvimento dos seguintes modelos de políticas:

- LeastPrivilegeModel voltado para a efetivação do princípio do privilégio mínimo. Sua realização demonstrou como utilizar a plataforma proposta para estabelecer uma política com granularidade própria e válida por toda uma VO;
- IDS_Control que interliga o ambiente de execução a um IDS em um sítio, aumentando a segurança no serviço de instanciação de tarefas. Sua construção mostrou a capacidade do GridMultiPolicy de integrar o serviço de autorização com mecanismos pré-existentes em sítios.

Longe de esgotar o assunto, o teste operacional contou com propósitos específicos. A flexibilidade da plataforma proposta, além de contribuir com a segurança das grades computacionais, abre possibilidades de estudar seu uso com diferentes políticas e mecanismos em diversos tipos de sistemas.

A escolha da tecnologia Java para a realização do ambiente de execução JMPE, se, por um lado, coaduna com os preceitos da grade computacional de heterogeneidade, permitindo sua utilização em diferentes sistemas operacionais, já por outro, é uma limitação, criando incompatibilidade com aplicações não Java. Trabalhos futuros podem ocupar-se na realização de uma solução mais abrangente permitindo o uso de aplicações desenvolvidas em diferentes linguagens.

Referencias

AXIS (2006), Java platform for creating and deploying web services applications, http://ws.apache.org/axis/, November.

Bavier A., Bowman, M. et all. (2004) “Operating System Support for Planetary-Scale Network Services” In *NSDI'04*, San Francisco, US.

Burruss, J.; Fredian, T.; Thompson, M. (2006) “ROAM: An Authorization Manager for Grids”. In: Journal of Grid Computing, Volume 4, Number 4, pp. 413-423(11).

Coetzee, M. and Eloff, J. (2003) “Virtual enterprise access control requirements” In: South African Institute of Computer Scientists and Information Technologists (SAICSIT 2003), Indaba.

Damiani, E,, Vimercati, S. e Samarati, P. (2002) “A Fine-Grained Access Control System for XML Documents”, In: ACM Transactions on Information and System Security (TISSEC), vol. 5, n. 2, pp. 169-202.

Damiani, E., Vimercati, S. e Samarati, P. (2005) "New Paradigms for Access Control in Open Environments," In: Proc. of the 5th IEEE International Symposium on Signal Processing and Information, Athens, Greece.

Debar, H., Curry, D., Feinstein, B. (2006) “RFC4765: The Intrusion Detection Message Exchange Format (IDMEF)”, IETF.

Debar, H., Dacier, M., Wespi, A. (1999) “Towards a taxonomy of intrusion-detection systems”, In: J. Computer and Telecommunications Networking, vol. 31, no. 9, pp. 805-822.

Foster, I. (2005) “Globus Toolkit Version 4: Software for Service-Oriented Systems” In: *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13.

Gridpmap (2005), International Grid Policy Management Authority. http://gridpma.org.

Humphrey, M., Thompson, M. and Jackson, K. (2005) “Security for Grids”, In: *IEEE*, vol. 93, no. 3, pp. 644-652.

Keahey, K., Ripeanu, M. and Doering K. (2004) “Dynamic Creation and Management of Runtime Environments in the Grid”. In: Workshop on Designing and Building Web Services (GGF 9), Chicago, IL.

Lang, B., Foster, I., Siebenlist, F., Ananthakrishnan, R., Freeman, T. (2006) A Multipolicy Authorization Framework for Grid Security. In: The Fifth IEEE International Symposium on Network Computing and Application,.

Lorch, M. and Kafura, D. (2004) “The PRIMA Grid Authorization System”. In: Journal of Grid Computing, Vol. 2, Num 3.

Mattes, L. and Zuffo, J. (2006) “Access Control Platform for Submitted Jobs in Communication”. In: Network, and Information Security (CNIS 2006), Cambridge.

Meinel C. (2005) “A Framework for Supporting Distributed Access Control Policies”. In: 10th IEEE Symposium on Computers and Communications (ISCC'05), pp. 442 - 447 .

Militelli, L. C. (2006) “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.

Otenko, S. and Chadwick, D. (2003) “A Comparison of the Akenti and PERMIS Authorization Infrastructures,” http://sec.isi.salford.ac.uk/download/AkentiPERMISDeskComparison2-1.pdf.

Park, S. and Humphrey M. (2006) “Authorizing Remote Job Execution based on Job Properties”. In 2th IEE international conference on e-science and grid computing (e-Science 2006), Amsterdam.

Park, S., Kim, K., Jang, J., Noh, B. (2003) "Supporting interoperability to heterogeneous IDS in secure networking framework" The 9th Asia-Pacific Conference on Communications, Volume 2, p. 844-848

Pearlman, L., Welch, V., Foster, I. and Kesselman, C. (2003) “The Community Authorization Service: Status and Future”. In: Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, USA.

Pistoia, M., Reller, F. e Gupta, D. (2005) “Java 2 Network Security”, Prentice-Hall, 2nd Edition.

Poppi, S. (2006) “Snort-IDMEF plug-in”, http://sourceforge.net/projects/snort-idmef/, March.

Prelude Hybrid IDS, (2006) “Prelude 0.9 Handbook”, https://trac.prelude-ids.org/wiki/PreludeHandbook, December.

Saltzer, J. H. e Schroeder, M. D. (1975) “The Protection of Information in Computer Systems”. Proceedings of the IEEE, 1975. 63(9): p. 12781308.

SAML (2007) Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) Version 2.0, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

Schulter, A., Reis, J. A., Koch, F., Westphall, C. B. (2006) “A Grid-based Intrusion Detection System”, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL’06)

Steenbakkers, M. (2003) “Guide to LCAS, Version 1.1.16”, 15 September 2003. Documentation of the European DataGrid Project.

Tham, C. K., Buyya, R. (2005) “SensorGrid: Integrating Sensor Networks and Grid Computing”, Special Issue on Grid Computing, Computer Society of India.

Vollbrecht, J et al. (2000) “AAA Framework” Internet RFC2904, Internet Engineering Task Force, Network Working Group, August.

Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, S. and Tuecke, S. (2003) “Security for Grid Services”. In Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press.

Welch, V., Barton, T., Keahey, K. and Siebenlist, F. (2005) “Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration”. In: Proceedings of the 4th Annual PKI R&D Workshop.

XACML (2007), Extensible Access Control Markup Language (XACML) Version 2.0, http://www.oasis-open.org/committees/xacml.