UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Lucas Marcus Bodnar

ADAPTAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO EM SISTEMA IMUNOLÓGICO ARTIFICIAL

Florianópolis

2014

Lucas Marcus Bodnar

ADAPTAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO EM SISTEMA IMUNOLÓGICO ARTIFICIAL

Trabalho de conclusão de curso submetido ao Curso de Bacharelado em Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. João Bosco Mangueira Sobral
Coorientador: Prof. Dr. Fernando Augusto da Silva Cruz

Florianópolis

2014

Lucas Marcus Bodnar

ADAPTAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO EM SISTEMA IMUNOLÓGICO ARTIFICIAL

Este Trabalho de conclusão de curso foi julgado aprovado para a obtenção do Título de "Bacharel em Sistemas de Informação", e aprovado em sua forma final pelo Curso de Bacharelado em Sistemas de Informação.

Florianópolis, 02 de julho 2014.

Prof. Dr. Leandro José Komosinski
Coordenador do Curso

Banca Examinadora:

Prof. Dr. João Bosco Mangueira Sobral
Orientador

Prof. Dr. Fernando Augusto da Silva Cruz
Coorientador

Prof. Dr. Renato Bobsin Machado - UNIOESTE/PR

Prof. Dr. Ricardo Azambuja Silveira

Dedico este trabalho a minha família, meus amigos e minha namorada, os quais me ajudaram a tornar esta conquitas possível.

AGRADECIMENTOS

Agradeço ao professor João Bosco Mangueira Sobral, meu orientador, por toda a ajuda e ensinamentos durante esta jornada.

Ao professor Fernando Augusto da Silva Cruz, pela chance que me deu deste trabalho.

Ao Renato Bobsin Machado, por permitir dar continuidade ao seu grande trabalho, e por toda a ajuda durante este caminho.

A minha família, que nunca abdicou de esforços pela minha educação.

A meus amigos e colegas de curso, que torceram por mim e me ajudaram a tornar possível mais esta etapa em minha vida.

A UFSC, seus professores e todas as pessoas envolvidas, que me ajudaram direta ou indiretamente, permitindo um ambiente propício aos estudos, transmitindo ensinamentos ou simplesmente me apoiando, tornando esta jornada exitosa.

"Faça da pedra de tropeço, um degrau de subida. Transforme cada fato negativo, em uma experiência positiva."

Bruce Lee

RESUMO

Neste trabalho é apresentado um estudo sobre sistemas de detecção de intrusão, sistemas imunológicos artificiais e agentes móveis baseado em uma dissertação onde foi desenvolvido um sistema imunológico artificial, o qual se encontrava fora de funcionamento por utilizar uma plataforma de agentes móveis obsoleta. Ao final deste estudo, foi possível obter um Sistema de Detecção de Intrusão novamente funcional, e com melhor desempenho, através da substituição da plataforma de agentes móveis para uma mais atual. Também houveram mudanças na ferramenta, as quais melhoraram a segurança, com autenticação, e a facilidade do uso, com algumas melhorias na iniciação da plataforma de agentes móveis, na interface de seleção de destinos e adaptação para diferentes Sistemas Operacionais.
Palavras-chave: IDS. SDI. Sistema de Detecção de Intrusão. Sistema Imunológico Artificial. Agentes Móveis. JADE.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Este trabalho apresenta um estudo sobre segurança de redes, mais especificamente, um estudo sobre sistemas imunológicos artificiais, onde é utilizado como base uma tese, a qual foi desenvolvida conjuntamente um Sistema de Detecção de Intrusão, utilizando agentes móveis (MACHADO, 2005). Este Sistema de Detecção de Intrusão desenvolvido em *Java* e construído originalmente sobre a plataforma de agentes móveis *Grasshopper*, que, entretanto, foi descontinuada quanto ao seu uso. O trabalho aqui apresentado, adapta uma nova plataforma de agentes móveis para este SDI, usando para os agentes estáticos e móveis a plataforma *JADE - Java Agent Development Environment*, no sentido de que este sistema de defesa possa ser realmente utilizado. A necessidade por eficácia em segurança computacional tem crescido em função do aumento considerável da ocorrência de ataques (CERT/CC, 2014). O problema cria um nicho para a pesquisa em segurança, tendo-se aqui aplicado método com inspiração no sistema imunológico humano (SIH). Este trabalho consiste em uma abordagem para detecção de intrusão inspirada nos conceitos oriundos do sistema imunológico humano (SIH) e aplicando a tecnologia de agentes móveis para a implementação de requisitos da solução.

1.1 MOTIVAÇÃO

O sistema em Machado (2005) foi desenvolvido conjuntamente uma dissertação de mestrado, e testado em dois grandes experimentos de redes reais, tais como uma rede local e num ambiente de provedor de Internet. Contudo, atualmente, com o advento da Computação em Nuvem e o aumento da importância dos dados, alguns desafios tem surgido com relação aos problemas de segurança desses ambientes, o que motivou a atualização do trabalho original, para uma plataforma mais moderna. Por serem implantadas como uma rede pública através da Internet, um ambiente de nuvem computacional, e outros ambientes que necessitem de segurança e estão ligados a internet, tem muitos riscos de ataques (CERT/CC, 2014), que podem originar intrusões no ambiente, e implicar em prejuízos sobre dados críticos, com impacto significativo para as organizações. Neste caso, este sistema de detecção de intrusão, poderá ser implantado nos *clusters* de uma nuvem computacional, ou mesmo em uma rede local, garantindo assim uma melhor

segurança destes ambientes. A busca por uma infra-estrutura de segurança, usando-se sistemas de detecção ou prevenção de intrusões, pretende garantir os requisitos de confidencialidade, privacidade, integridade, disponibilidade e autenticação. Baseado nessas características tem-se aplicado diversidade de métodos, entre as quais: sistemas especialistas, redes neurais artificiais, mineração de dados, agentes móveis e sistemas imunológicos artificiais.

1.2 JUSTIFICATIVA

Apesar da segurança que normalmente é implantada em ambientes computacionais, vulnerabilidades ainda podem acontecer, e é necessário responder aos incidentes nesses ambientes, os quais costumam ser complexos. A utilização de um sistema de detecção de intrusão, adiciona mais um nível de segurança, o que será o último nível de defesa, que conterá os ataques destinados a intrusões (ataques bem sucedidos), onde entram os sistemas de detecção de intrusões.

Uma outra justificativa importante, diz respeito ao ensino de disciplina de segurança de redes. A implantação em laboratório do sistema de detecção de intrusão adaptado neste trabalho, pode contribuir para melhoria do ensino prático para mostrar a estrutura e os conceitos de um sistema de detecção de intrusão em funcionamento.

1.3 OBJETIVOS

O principal objetivo deste trabalho prevê um estudo sobre Sistemas de Detecção de Intrusão baseados em Sistemas Imunológicos Artificias (SIA). Também é feito um breve estudo sobre agentes móveis, os quais são utilizados como principais componentes no SIA, representado as células do corpo humano, assim sendo possível, ao final do trabalho, ter um SDI funcional para testes, utilização e estudos.

1.3.1 Objetivos Específicos

- Estudar ferramentas de detecção de intrusão baseada em sistemas imunológicos artificiais.

- Estudar as plataformas de agentes móveis existentes e encontrar a mais apropriada para a substituição no SDI.

- Avaliar o desempenho do sistema de detecção de intrusão, comparando-o com os resultados obtidos sobre o funcionamento de um sistema existente.

No trabalho original, adotou-se o método de detecção baseado em anomalias, arquitetura centralizada e baseada em *host* e respostas passivas por meio do envio de *e-mails* aos administradores. Para a geração de eventos foram utilizados os geradores de *logs Syslog* e *Syslog-ng* e para a tarefa de análise foi aplicada a ferramenta *Logcheck*. A solução foi baseada na monitoração e análise sequencial de *logs*, universalidade do ambiente operacional *UNIX*, reconhecimento de anomalias e envio de alarmes ao administrador.

1.4 ESTRUTURA DO TRABALHO

Nos Capítulos 2, 3, 4 e 5, é feito a revisão bibliográfica, onde é mostrado um estudo sobre os conceitos necessários para o desenvolvimento e entendimento deste trabalho, e nos Capítulos 6 e 7 é demonstrado a substituição da plataforma e os testes referentes ao novo sistema de detecção de intrusão adaptado. A abordagem de cada capítulo é definido adiante.

No Capítulo 2 são apresentados os conceitos relativos ao sistema imunológico humano e aos sistemas imunológicos artificiais, incluindo sua anatomia, arquiteturas, processos de detecção e reatividade, propriedades e princípios.

O Capítulo 3 é destinado a apresentar o problema da segurança de redes e os métodos aplicáveis para detecção de intrusão.

No Capítulo 4 é definida a tecnologia de agentes móveis, citando propriedades e características que a diferenciam entre os distintos paradigmas aplicados em sistemas distribuídos.

No Capítulo 5 apresenta-se a abordagem de detecção de intrusão deste trabalho. Na primeira parte do capítulo define-se o projeto computacional, mencionando configurações, tecnologias utilizadas e suas particularidades. A segunda parte do capítulo contém a definição do modelo imunológico artificial, onde apresenta-se a arquitetura imunológica que motivou a escolha de cada componente descrito no modelo computacional.

O Capítulo 6 mostra o motivo da escolha da plataforma, um breve resumo da plataforma antiga, o *Grasshopper* (IKV, 1999), e da plataforma nova, o *JADE* (BELLIFEMINE; CAIRE; GREENWOOD, 2007), e também as principais mudanças no sistema de detecção de intrusão

(SDI).

O Capítulo 7 é destinado à avaliação de desempenho em função das características da plataforma *JADE*, comparada à plataforma *Grasshopper*.

No Capítulo 8 são apresentadas algumas as conclusões importantes e a proposta de trabalhos futuros.

Para a elaboração do texto aplicou-se algumas padronizações. Assim, termos em inglês são apresentados em *itálico*, terminologias da áreas biológica são ressaltados em fonte `verbatim` e ferramentas computacionais são destacadas com fonte *emphasise*.

2 SISTEMAS IMUNOLÓGICOS

Para o trabalho original (MACHADO, 2005), foi aplicado em um Sistema de Detecção de Intrusão o conceito de Sistemas Imunológicos Artificiais baseado em Agentes Móveis, utilizando assim pequenos fragmentos de programas como células percorrendo por todo o organismo, ou seja, a rede. Estes conceitos estão mais detalhadamente explicados na dissertação Machado (2005).

2.1 SISTEMA IMUNOLÓGICO HUMANO

O conceito de sistema imunológico humano (SIH), o qual foi baseado o SDI, consiste em um conjunto de órgãos, células e moléculas responsáveis pela defesa do corpo contra ataques de invasores externos, tais como bactérias, vírus, fungos ou parasitas (MACHADO, 2005). O SIH é composto por várias camadas, cada qual com uma função específica, e as principais partes deste sistema são:

a) As barreiras físicas, que são os primeiros obstáculos contra os organismos causadores de doenças e são compostas pela pele e pelo sistema respiratório, que podem conter `macrófagos` e `anticorpos`.

b) As barreiras bioquímicas, que são os fluídos do corpo humano, como suor, saliva, lágrimas, ácidos estomacais, pH e temperatura corporal, que podem eliminar ou criar um ambiente desfavorável para diversos tipos de invasores.

c) E o sistema imunológico inato e adaptativo, onde o inato tem uma capacidade limitada em reconhecer e combater os agentes `patogênicos`, ou seja, aplicando as mesmas técnicas de detecção e resposta a todos os organismos causadores de doenças. Já o adaptativo convém da capacidade de identificar o agente `patógeno`, assim memorizando e produzindo respostas especializadas.

A Figura 1 ilustra essas barreiras classificadas por Hofmeyr (2000).

O SIH produz estas ações de defesa a partir de suas células, cada qual responsável por uma atividade dentro do sistema, mas todas trabalhando em conjunto para manter o corpo humano seguro. A divisão das células dentro do SIH é representada pela Figura 2.

Para este trabalho, será explicado somente as células do sistema adaptativo, e os principais componentes que auxiliam na imunidade,

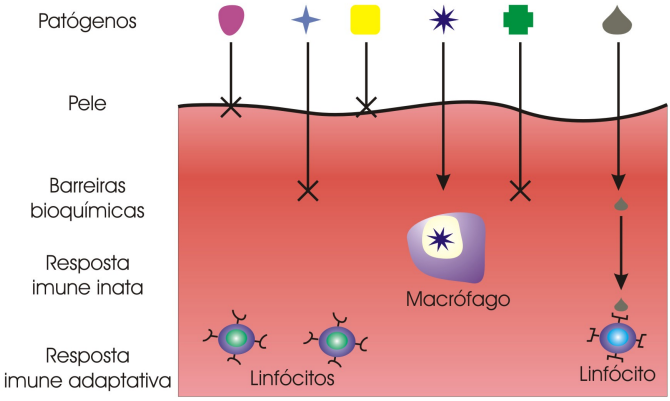

Figura 1 – Barreiras do Sistema Imunológico Humano. (MACHADO, 2005).

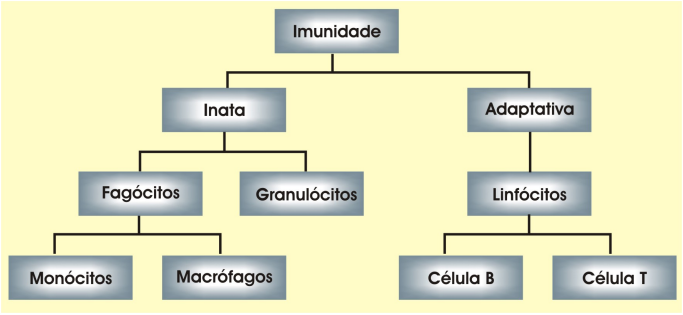

Figura 2 – Hierarquia de Células do Sistema Imunológico Humano. (MACHADO, 2005).

abstraindo e adicionando conceitos importantes que estejam ligados a este trabalho. Assim temos:

- `Linfócitos`: São pequenas células de `glóbulos brancos`, sua função é de especificar as atividades do sistema imunológico. Dois dos principais tipos de `linfócitos` são as `células B` e as `células T`.

- `Células B`: São células criadas na medula óssea (do inglês Bone Narrow, por isto o B), onde após sua ativação, pode sofrer diferenciação para `plasmócitos`.

- Células T: São células maturadas no Timo (por isto o T) que funcionam no reconhecimento de antígenos, e atuam diretamente nas células do corpo atacadas por viroses, parasitas ou fungos.
- Anticorpos: São moléculas protéicas solúveis produzidas em resposta a um antígeno específico pela célula B.
- Plasmócitos ou Linfócitos Plasma: São células em maior quantidade nos locais onde há maior risco de infecção por bactérias ou onde há inflamação crônica. São provenientes das células B e responsáveis pela criação dos anticorpos.
- Células Tronco: São células com a capacidade de se autorreplicar, assim criando outras células, como os Linfócitos B ou T.

Com isto temos os principais componentes do sistema imunológico humano, e suas principais funções, onde o complexo sistema permite uma excelente proteção contra agentes externos, garantindo assim a saúde do corpo humano. Portanto, esses conceitos podem ser transportados para as mais diversas áreas que exigem segurança de um sistema, inclusive redes de computadores e ambientes de nuvem computacional.

2.2 SISTEMAS IMUNOLÓGICOS ARTIFICIAIS - SIA

O sistema Imunológico Artificial, como o nome já diz, é baseado no Sistema Imunológico Biológico, geralmente no Humano, onde buscase inspiração na natureza e seus processos para resolver problemas de diversas áreas. Não é de hoje que o homem vem buscando inspiração na natureza, e nem é exclusividade da área de softwares. No campo artístico, a natureza sempre foi fonte de inspiração para os mais diversos artistas, já nos meios de transporte utilizados, o avião, com sua extrema importância atualmente, somente foi possível construir depois de vários estudos de voos de aves. Mesmo coisas mais simples, como o velcro, foram obtidos depois de observar alguns eventos da natureza, que neste caso, como algumas plantas grudavam insistentemente nas roupas. Atualmente, muitos dos objetos e tecnologias que utilizamos no dia a dia, são de ideias obtidas de observações dos eventos da natureza e dos seres pertencentes a ela. Já na área de computação, temos os conceitos de redes neurais artificiais, computação evolutiva, computação molecular, e entre elas, os Sistemas Imunológicos Artificias, sendo neste último, o conceito no qual o trabalho original foi baseado.

Entre os vários conceitos de SIA, o trabalho utiliza a seguinte ideia, formulado por Dasgupta e Forrest (1999):

> Os sistemas imunológicos artificiais são compostos por metodologias inteligentes, inspiradas no sistema imunológico biológico, para a solução de problemas do mundo real. (DASGUPTA; FORREST, 1999).

Com isto, o trabalho original buscou inspiração no sistema imunológico humano para construir um Sistema de Detecção de Intrusão (SDI), onde utiliza como células uma plataforma de agentes móveis, que será explicada mais à frente. Portanto, como conceito final do sistema imunológico adaptativo temos os `linfócitos` do SDI e os *patches* e atualizações funcionando como analogia as vacinas.

3 SEGURANÇA DE REDES E DETECÇÃO DE INTRUSÃO

Neste capítulo são apresentados conceitos referentes à segurança de redes de computadores e detecção de intrusão, técnicas e classificações de invasões, métodos e arquiteturas aplicados em sistemas de detecção de intrusão (SDI) e possíveis implementações de respostas computacionais.

3.1 AMEAÇAS À SEGURANÇA

Os sistemas computacionais atuais estão sujeitos a um extensivo conjunto de ataques em função da exploração de distintas vulnerabilidades, estendendo-se desde tentativas de negação de serviços até sofisticados ataques aplicando recursos distribuídos.

Uma ameaça consiste na possibilidade de violação de um sistema. De forma geral, as principais classes de violações são descritas por Stallings (2003), e suas arquiteturas são apresentadas na Figura 3.

a) **Interrupção**: O fluxo de mensagem é interrompido, impossibilitando que estas cheguem até seu destino. Afeta a *disponibilidade* dos recursos.

b) **Interceptação**: Acesso não autorizado às informações confidenciais. Um exemplo é a captura de dados na rede ou a cópia ilegal de um arquivo. Este é um ataque a *confidencialidade*.

c) **Modificação**: Forma de acesso não autorizado em que as mensagens são interceptadas, alteradas e reenviadas ao destinatário. Esse ataque afeta a *confidencialidade* e a *integridade* da mensagem.

d) **Fabricação**: Esse tipo de ataque caracteriza-se pela inserção de informações nos sistemas, constituindo uma violação à *autenticidade* das informações.

Os ataques exploram vulnerabilidades e violam as políticas de segurança. A busca por uma proteção contra essas possíveis invasões constitui uma metodologia que pode incluir aplicações criptografadas (STALLINGS, 2003), métodos de autenticação e validação, instalação de *patches* do sistema operacional e dos serviços, aplicação de *firewalls*

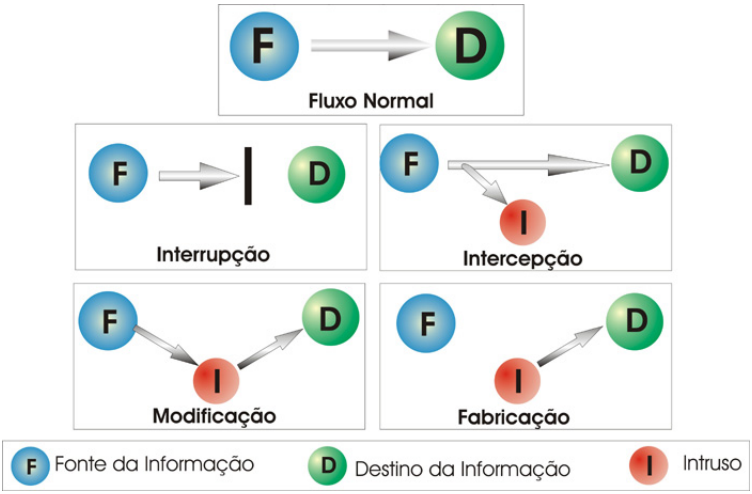

Figura 3 – Ameaças à Segurança. Fonte: (STALLINGS, 2003).

e, como uma última barreira, a inclusão de sistema de detecção de intrusão. O conjunto de todas essas medidas determina a política de segurança de uma organização.

3.1.1 Classes de Ataques

Um ataque pode ser definido como uma ação maliciosa que viola as políticas de segurança e compromete a integridade ou a disponibilidade dos recursos em um sistema. Possui como princípio elementar a exploração de vulnerabilidades, tanto de sistemas como de protocolos existentes. Os resultados de um ataque podem caracterizar uma invasão ou acarretar indisponibilidade dos serviços providos (BARBOSA; MORAES, 2000).

Com base nessa definição, os ataques podem ser classificados em função de suas propriedades, tais como as definidas por Allen et al. (2000). Esses ataques são divididos em três classes: ataques de sondagem (*Scanning Attacks*), comprometimento de recurso (*Denial of Service Attacks - DoS*) e penetração em sistemas (*Buffer Overflows*) (TAVARES, 2002).

ATAQUES DE SONDAGEM

Constituem métodos investigativos que baseiam-se na busca por alvos, na rede ou em sistemas, por meio do envio de pacotes distintos. As respostas recebidas permitem ao atacante aprender sobre as características do sistema a ser atacado e suas vulnerabilidades. Esse procedimento possibilita a extração de informações como: tipos de tráfego, endereços de servidores, serviços ativos, sistemas operacionais utilizados, versões dos serviços, entre outras. De posse desses dados, o atacante pode utilizar ataques e ferramentas que explorem as vulnerabilidades encontradas (TAVARES, 2002).

Algumas categorias de ataques de sondagem incluem: *hosts* quando exploram dados relativos a um servidor, seus serviços e *OSfingerprinting* nas circunstâncias em que o objetivo é a identificação de características do sistema operacional (MACHADO, 2005).

ATAQUES DE COMPROMETIMENTO DE RECURSOS

Os ataques de comprometimento de recursos, *Denial of Service Atacks (DoS)*, são projetados para interromper ou negar completamente o acesso a redes, *hosts*, serviços e recursos. Afetando diretamente a disponibilidade do sistema.

Existem dois tipos principais de ataques *DoS*:

a) Exploração de Falha: Técnica que explora falhas nos serviços do alvo com o objetivo de provocar o esgotamento dos recursos.

b) Inundação: Método intrusivo que envia pacotes a taxas que os sistemas não tenham capacidade de processar.

Uma variação do método de comprometimento de recursos é o DoS Distribuído (*DDoS*), que resulta da manipulação conjunta de vários *sites* por um invasor, mantendo o objetivo de sobrecarregar o alvo. Essa técnica surgiu da necessidade dos invasores de atacarem servidores com maior capacidade de processamento (TAVARES, 2002).

ATAQUES DE PENETRAÇÃO

Tavares (2002) conceitua ataques de penetração como o envolvimento na aquisição e/ou alteração de privilégios, recursos ou dados. Em um comparativo com os ataques *DoS* e Sondagem, esses são mais

desastrosos e podem comprometer a disponibilidade, integridade e controle de sistemas.

Essa categoria de intrusão tem por objetivo obter o controle de um sistema por intermédio da exploração de uma grande variedade de falhas de *software*. As vulnerabilidades mais exploradas são estouro de *buffer*, de pilha e de inteiros.

3.2 DETECÇÃO DE INTRUSÃO

Conforme Crosbie e Spafford (1995), uma intrusão pode ser definida como um conjunto de ações que tentam comprometer a integridade, confidencialidade ou disponibilidade de recursos.

A área de pesquisa denominada *Detecção de Intrusão* desenvolveu-se com o objetivo de aplicar técnicas computacionais para a identificação de violações ocorridas e implementação de contramedidas.

Existem diversos mecanismos de segurança, no entanto, há carência por métodos eficazes e automáticos. Abordagens computacionais aplicadas a estes problemas são denominadas Sistemas de Detecção de Intrusão (SDIs).

Os SDIs são inseridos como a última linha de defesa dentro de uma arquitetura computacional, o que os torna de importância vital, possibilitando inferir sobre a legitimidade de ações realizadas e possuindo comportamento proativo em situações de ataque (BERNARDES, 1999).

Essa seção é dedicada ao estudo de SDI, abrangendo conceitos relativos à sua padronização, classificação e linhas de pesquisa correlatas.

3.2.1 Classificação dos Sistemas de Detecção de Intrusão

Um SDI é composto por uma arquitetura de *hardware* e *software* responsáveis pela detecção de intrusão. As abordagens adotadas para esses sistemas podem ser classificadas em função de quatro critérios: método de detecção, arquitetura, frequência de uso e comportamento após a detecção (CAMPELLO; WEBER, 2001).

MÉTODOS DE DETECÇÃO DE INTRUSÃO

Os métodos de detecção permitem classificar os SDIs conforme a abordagem aplicada para o problema. Nessa classe distinguem-se a detecção por anomalia, detecção por abuso e detecção híbrida.

a) Detecção por Anomalia

Essa abordagem baseia-se no estudo comportamental. Para isso observa-se as variações na utilização dos serviços oferecidos por um sistema computacional ou *host*, buscando-se perfis que fujam significativamente ao comportamento normal estabelecido. Dessa forma, é necessário determinar o conjunto de atividades normais de um sistema. Assume-se o princípio que toda atividade intrusa é necessariamente anômala.

Para a determinação de estados anômalos são aplicadas diferentes técnicas, as quais são logicamente classificadas por Axelsson (2000) e incluem atividades como detecção por limiar de atributos (*threshold*), distribuição de atributos seguindo medidas estatísticas paramétricas, técnicas baseadas em regras, modelagem baseada em estados, algoritmos genéticos, redes neurais artificiais, entre outras. Uma técnica comportamental que está sendo estudada são os sistemas imunológicos artificiais, os quais se inspiram na detecção de `patogenias` realizada pelo SIH.

Essas definições de conjuntos normais e anômalos dão margem a geração de alarmes falsos. A seguir define-se as possíveis categorias de eventos gerados por essa técnica:

Falsos Negativos: São eventos intrusos mas considerados pelo sistema de detecção de intrusão como não anômalos. Sua detecção falha e o sistema reporta a ausência da intrusão. Esse é o pior caso e deixa o sistema vulnerável.

Falsos Positivos: Essa categoria inclui os eventos não intrusos, porém classificados pelo sistema de detecção de intrusão como anômalos, caracterizando uma situação indesejada. Não se trata de um ataque, mas o sistema o classifica como tal.

Verdadeiros Negativos: Eventos não intrusos e não anômalos. Não são reportados pelo sistema.

Verdadeiros Positivos: Eventos intrusos e anômalos que são devidamente reportados e tratados pelo sistema de detecção de intrusão.

b) Detecção por Abuso

A detecção por abuso é baseada na análise das atividades do sistema. Para isso é realizada uma busca por eventos ou conjunto de eventos pré-definidos e conhecidos, normalmente chamados de assinaturas (AXELSSON, 2000). Essas características permitem a detecção eficaz e rápida de padrões conhecidos, o que minimiza a ocorrência de alarmes falsos.

Essa técnica possui como limitação (CANSIAN, 1997) a possibilidade de detectar somente ataques conhecidos, tornando-se ineficaz para novos padrões de ataques ou variações de uma assinatura conhecida. Como consequência é necessária a atualização contínua da base de dados armazenando as assinaturas.

c) Detecção Híbrida

Os métodos de detecção por anomalia e abuso possuem características distintas, com vantagens e desvantagens. Cada uma das abordagens se torna mais adequada para certos tipos de ataques e, por isso, muitas propostas estão utilizando uma técnica híbrida com o intuito de tornar a detecção mais eficiente e com melhor desempenho (BERNARDES, 1999).

ARQUITETURAS DE DETECÇÃO DE INTRUSÃO SEGUNDO O ALVO

A arquitetura de um SDI reflete em seu desempenho, podendo-se utilizar modelos simples ou combinados. Com relação ao alvo, a arquitetura pode ser baseada em *Host*, em Rede ou Híbrida.

a) Detecção de Intrusão Baseada em Host

Um SDI baseado em *Host* é chamado *Host Based Intrusion Detection System* (HIDS). Com essa abordagem, todos os componentes, desde a coleta até a gerência, estão localizados no mesmo *host*. As informações são coletadas na máquina local e os eventos podem ser originados a partir de arquivos de auditoria, *logs* do sistema, dados sobre usuários, serviços e processos ou mesmo pacotes de rede relacionados a atividades locais. Essa abordagem possibilita a independência da rede, a detecção de ataques internos e a facilidade para a geração de ações reativas. Como desvantagens dessa solução citam-se: a dificuldade em tratar ataques da rede, ataques ao próprio SDI e dificuldade de configuração e manutenção.

b) Detecção de Intrusão Baseada em Rede

Os SDIs baseados em rede são denominados *Network Based Intrusion Detection System* (NIDS). A aquisição de dados é realizada capturando o tráfego da rede em modo promíscuo, de forma que os sensores são alocados em posições estratégicas nos diversos segmentos de rede. A monitoração não ocorre somente no *host*, mas em todo o tráfego do segmento. Os pacotes capturados podem ser pré-processados para seleção e extração de informações relevantes.

Uma das dificuldades reside em determinar os melhores locais para o posicionamento dos sensores. Estes devem ser alocados em posições estratégicas nos diversos segmentos da rede. Um outro problema se concentra na utilização de *switches* como equipamento de interconexão, devido a ausência de *broadcast*.

Com relação aos serviços e aplicações, a utilização de criptografia incapacita o SDI a realizar análises, devido a impossibilidade de reconhecimento de assinaturas ou padrões de ataques em dados criptografados.

Entre as principais vantagens dessa abordagem cita-se: detecção de ataques externos, facilidade de utilização, desempenho e independência de plataforma. Como fatores negativos, essa técnica apresenta dificuldade para o tratamento de grande volume de dados, é dependente da rede e possui dificuldade em executar processos reativos quando existem ataques em andamento.

c) Detecção de Intrusão Híbrida

SDIs híbridos combinam as principais vantagens das técnicas baseadas em *Host* e em Rede, monitorando tanto sistemas locais, quanto o tráfego da rede. O objetivo dessa abordagem é chegar a um ponto de equilíbrio entre desempenho, simplicidade, abrangência e robustez.

ARQUITETURAS DE DETECÇÃO DE INTRUSÃO SEGUNDO O LOCAL

Além da classificação dos SDIs em função do alvo, a localização está relacionada à forma como seus componentes funcionais encontram-se arranjados. Esse conceito permite classificar os sistemas de detecção em centralizados, hierárquicos (parcialmente distribuídos) ou distribuídos (CAMPELLO; WEBER, 2001).

Nos sistemas centralizados todos os seus componentes localizam-se no mesmo ponto, desde a geração dos eventos até a configuração e gerência.

A arquitetura hierárquica representa a classe de sistemas que estão parcialmente distribuídos, possuindo fortes relações hierárquicas entre si. Nesse caso, embora alguns elementos sejam distribuídos, as tarefas de análise e tomada de decisões ficam concentradas em um único local.

Em SDIs distribuídos, seus elementos e funções estão em pontos diversificados com relações mínimas de hierarquia entre eles. Essa classe de SDI pode ter grupos de detectores, analisadores, armazenamento e respostas em locais distintos. De forma análoga, cada função pode ser constituída de múltiplas unidades trabalhando cooperativamente.

COMPORTAMENTO PÓS-DETECÇÃO E FREQUÊNCIA DE USO

A frequência de uso em SDIs é vinculada ao intervalo de tempo de ativação dos processos de aquisição e análise. Os SDIs classificados em tempo contínuo ficam constantemente em execução, ao passo que SDIs que atuam em intervalos de tempo executam essas funções em períodos de tempo definidos.

A última classificação dos SDIs é em função da geração de respostas após a detecção de eventos intrusivos. Conceitua-se como respostas passivas aquelas que emitem algum alerta ao administrador sobre as ocorrências, sendo dependente da intervenção humana. A outra categoria, respostas ativas, efetua respostas automáticas.

3.2.2 Sistemas Imunológicos Artificiais Aplicados à Segurança de Redes

O problema associado à segurança de redes estimulou a pesquisa de diferentes técnicas de detecção de intrusão.

A analogia entre problemas de segurança e processos biológicos foi reconhecida em 1987 com a introdução do termo *vírus de computador* (COHEN, 1987). A conexão entre imunologia e segurança de computadores foi estabelecida em 1994 com as publicações de Forrest e Perelson (1994) e Kephart (1994).

Conforme Somayaji, Hofmeyr e Forrest (1998), os sistemas imunológicos naturais constituem uma rica fonte de inspiração para a se-

gurança de redes. Entre as principais características desses sistemas aplica-se importantes princípios, tais como distribuição, diversidade, disponibilidade, adaptabilidade, autonomia, múltiplas camadas, detecção por anomalia e por abuso, entre outros.

A partir desses trabalhos pioneiros se desencadearam uma série de pesquisas abordando conceitos de sistemas imunológicos. Nesta linha de pesquisa, temos alguns trabalhos no Laboratório de Computação Móvel Distribuída e Segurança de Redes (Distributed Mobile Computing and Network Security, DMC-NS) da INE/UFSC no período de 2001 a 2007.

Em 2001 foi desenvolvida a dissertação de mestrado sobre um sistema de detecção de intrusão baseado no sistema imunológico humano com analise dos registros de atividades (JUCÁ, 2001) e em 2002 o primeiro trabalho foi publicado sobre essa dissertação (JUCÁ, 2002). Em 2003 mais um artigo foi publicado sobre anomalias e operações de detecção de intrusões (JUCÁ, 2003).

Em 2005 foi defendida a dissertação base deste trabalho, sobre detecção de intrusão baseado em sistema imunológico artificial e agentes móveis (MACHADO, 2005). Também em 2005 foi publicado um artigo visando operações de redes de computadores sobre detecção de intrusão com agentes móveis (MACHADO, 2005a). Ainda em 2005 foi publicado um capítulo de livro relativo a algoritmos inspirados na biologia (NOTARE, 2005). Mais tarde, em 2007, um artigo foi publicado sobre modelo de segurança para operações de redes abordando detecção de intrusão em tempo real, inspirados biologicamente (BOUKERCHE, 2007). No segmento dessa linha de pesquisa, está a proposta deste trabalho de atualizar o sistema de detecção de intrusão baseado no conceito de SIH e agentes móveis.

4 SISTEMAS DE AGENTES MÓVEIS

Nos últimos anos a consolidação da tecnologia de orientação a objetos estimulou a definição de novos paradigmas envolvendo a mobilidade de objetos. A tecnologia de agentes vem sendo aplicada em diferentes arquiteturas e em diversos campos, tais como: sistemas distribuídos, inteligência artificial e engenharia de software.

Em função dessas distintas abordagens não existe um consenso com relação a uma definição formal de agentes. No entanto algumas propriedades foram assumidas, permitindo determinar que um agente é um objeto que detém autonomia, inteligência e desempenha suas tarefas de acordo com o interesse de seus usuários. Para isso um agente pode comunicar-se com outros agentes e domínios, tendo como objetivo o cumprimento de suas tarefas. Conceitos como infraestrutura de agentes móveis são melhores descritos em Machado (2005).

4.1 AGENTES MÓVEIS

A tecnologia de agentes móveis é uma alternativa à convencional arquitetura Cliente/Servidor, combinando interação local com mobilidade de código.

Um agente móvel realiza tarefas de forma autônoma para atender usuários, e pode migrar para servidores que forneçam recursos necessários para a realização de uma atividade delegada. Servidores de agentes fornecem o ambiente de execução e diversos serviços para migração, comunicação e acesso a recursos.

Os agentes móveis são constituídos por código, estado e atributos. O código define o seu comportamento e deve ser escrito em uma linguagem interpretada. Após a mobilidade, o estado do agente é utilizado para permitir o seu retorno ao ponto onde havia parado. Os atributos são aplicados para descrever o agente para seus servidores, incluindo: um identificador único, um endereço para envio de resultados, tempo e história do agente. Os atributos são utilizados também para determinar limitações à mobilidade, isto é, limites de domínios onde pode trafegar.

Para que os agentes possam realizar suas tarefas, é necessária a existência de servidores de agentes, os quais lhes permitem interagir, comunicar-se com outros agentes e mover-se. Outra função do sistema de agentes consiste na garantia de segurança dos servidores e dos agen-

tes nele hospedados.

Da mesma forma que agentes de software, não existe consenso quanto a uma definição formal de agentes móveis. É assumida a definição de Chess, Harrison e Lebine (1995).

> "Agentes itinerantes são programas que são despachados de um computador de origem, viajam entre servidores em uma rede até que se tornem hábeis para completar a sua tarefa; eles movem processos que progressivamente executam tarefas se movendo de um lugar para outro" (CHESS; HARRISON; LEBINE, 1995).

A tecnologia de agentes móveis congrega características desejáveis e representam uma evolução nos conceitos de sistemas distribuídos. Dentro desse contexto, é descrito algumas vantagens desse paradigma de construção de *software*: Redução do tráfego de rede, ocultação da latência da rede, encapsulamento do protocolo, execução assíncrona e autônoma, adaptação dinâmica, independência de plataforma, robustez e tolerância a falhas.

Além do conceito de mobilidade (CHESS; HARRISON; LEBINE, 1995), agentes móveis agregam características inerentes ao conceito de multiagentes. Dessa forma, propriedades como cooperação, autonomia e representatividade foram herdadas da sua própria origem e outras foram acopladas a fim de suprir as necessidades exigidas para o funcionamento de modelos que utilizam esse paradigma. Entre elas citam-se (BERNARDES, 1999):

a) **Objetos Passantes**: quando um agente móvel é transferido, todo o objeto é movido, incluindo código, dados, itinerário e estado de execução.

b) **Assincronismo**: O agente móvel possui sua própria *thread* de execução, que pode ser executada tanto de forma síncrona como assíncrona.

c) **Interação Local**: O agente móvel pode interagir com outros agentes móveis ou objetos estacionários locais.

d) **Operações sem Conexão**: O agente móvel pode executar tarefas mesmo com a conexão fechada. Para a realização de transferências aguarda-se até que a conexão seja restabelecida.

e) **Execução Paralela**: Múltiplos agentes podem ser movidos para distintos servidores para a execução de tarefas paralelas.

4.1.1 Segurança de Agentes Móveis

A segurança de agentes móveis é um fator essencial para o sucesso da tecnologia, pois garantir a autenticidade, integridade e confidencialidade dos agentes e dos dados por eles transportados é subsídio para qualquer tipo de aplicação.

Para garantir a segurança de agentes móveis é necessário proteger o próprio agente, o *host* e a rede. A seguir são descritas as classes de ameaças a esses componentes.

a) **Ameaças ao Agente**: esse tipo de ameaça envolve a visita de agentes a *hosts* não confiáveis, os quais podem tentar extrair suas informações privadas, ou realizar ataques como falsificação, execução e acesso ilegal. Outra categoria de ataque que o agente pode sofrer é por parte de outro agente, abrangendo tentativa de extração de informações ou interrupção de sua execução. Uma terceira forma de ataque ao agente envolve fatores externos não autorizados, em que uma entidade pode alterar mensagens enviadas entre *hosts* ou escutar a comunicação entre eles e desvendar o conteúdo do agente.

b) **Ameaças ao *Host***: Um agente, ao visitar o servidor, pode tentar acessar ou corromper os arquivos ou interromper o servidor. Nessa categoria de ataques pode-se incluir acesso ilegal, disfarce, cavalo de troia, recusa de serviço e repúdio. Outra forma de violação contra *host* consiste no envio de *spam* por parte de uma entidade externa mal intencionada para tirá-lo de serviço.

c) **Ameaças à Rede**: Agentes podem multiplicar-se e mover-se intensivamente para congestionar a rede, caracterizando um ataque por recusa de serviços.

A maioria desses ataques pode ser evitada, para agentes móveis, por meio das técnicas de segurança existentes, tais como criptografia. Este requisito é um fator determinante para a escolha de uma plataforma de agentes móveis, a qual deve prover autenticação, confidencialidade, integridade, autorização, não-repúdio e auditoria.

As técnicas de criptografia e assinatura digital têm contribuído muito para a construção de ambientes que atendam a essas especificações de segurança.

4.1.2 Padronização de Agentes Móveis

A interoperabilidade entre sistemas de diferentes fabricantes é essencial para satisfazer as necessidades dos sistemas baseados em agentes móveis. Em função das diferenças arquiteturais e de implementação das plataformas de agentes, surgiram duas padronizações para agentes móveis, o FIPA (Foundation for Intelligent Physical Agents), hoje mantido pela IEEE Computer Society, e o MASIF (Mobile Agent System Interoperability Facility), da Object Management Group (OMG). Como o *JADE*, plataforma de agentes móveis utilizada nesse trabalho, utiliza o padrão FIPA, será feito um breve resumo a seguir.

O FIPA surgiu a partir de companhias e organizações que definiram um padrão para tecnologias de agentes genéricas, onde isto é feito através de uma série de documentos, definindo regras para o agente poder existir, operar e ser gerenciado. O FIPA contém três agentes especiais, o AMS (Agent Management System), o DF (Directory Facilitator) e o ACC (Agent Communication Channel), os quais gerenciam a plataforma, e descrevem a linguagem de conteúdo para o gerenciamento de agentes e ontologias. Este modelo definido pela FIPA é mostrado na Figura 4.

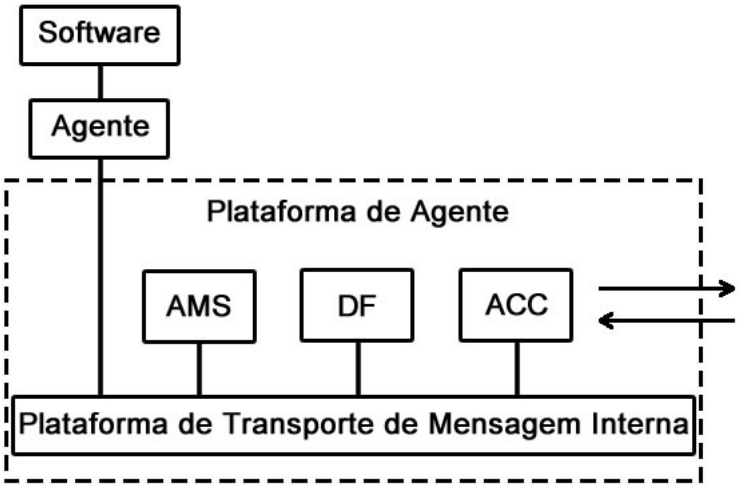

Figura 4 – Modelo de referência para plataformas de agentes definido pela FIPA. Adaptado de (TEIXEIRA, 2010).

Pela equipe do projeto da plataforma *JADE* estar presente em

todos os eventos realizados pela FIPA, além de ter participado de vários processos de padronização que originaram diversos documentos de especificação, a plataforma é capaz de atender a todos os requerimentos da FIPA. Com isto, no próximo capítulo é demonstrado a junção dos conceitos até aqui vistos para construir um conceito de modelo de detecção de intrusão, aplicando sistemas imunológicos artificiais e agentes móveis.

5 MODELO DE DETECÇÃO DE INTRUSÃO APLICANDO SISTEMAS IMUNOLÓGICOS ARTIFICIAIS E AGENTES MÓVEIS

A proposta do trabalho original (MACHADO, 2005) é inspirada pela anatomia, arquitetura e processos vinculados ao SIH e definiu um modelo para detecção de intrusão. Entre os requisitos projetados cita-se a identificação de anomalias, memorização, distribuição dos resultados analisados e geração de respostas específicas.

Para este capítulo foi considerado o conceito do trabalho original, contudo, foi feito modificações quanto a padronização dos agentes (MASIF para FIPA) para se adequar à nova plataforma de agentes móveis utilizadas.

5.1 MODELO COMPUTACIONAL

O modelo computacional estabelecido corresponde a definição de um sistema de detecção de intrusão baseado na abstração de algumas propriedades e processos do SIH e utiliza uma proposta arquitetural definida para sistemas imunológicos artificiais aplicados à segurança de redes (SOMAYAJI; HOFMEYR; FORREST, 1998). A analogia com cada componente do SIH contribuiu para a identificação de alternativas tecnológicas destinadas a uma solução com características genéricas. A presente abordagem foi definida buscando-se a implementação de requisitos como flexibilidade, adaptabilidade a distintas arquiteturas de *hardware* e *software*, políticas de segurança e objetivos de monitoração.

O SDI modelado trabalha com análise sequencial de registros de *logs* de auditoria e pode ser classificado como uma solução baseada em *host*, aplicando o método de detecção por *anomalia*, arquitetura distribuída, executada em tempo contínuo e gerando respostas ativas e passivas. A seguir, detalha-se cada uma dessas propriedades:

a) Método de Detecção por Anomalia: Essa técnica foi adotada fundamentada na observação de variações na utilização dos serviços monitorados. Para essa finalidade aplicou-se uma ferramenta de auditoria de *logs* (*Logcheck*. Essa ferramenta permite definir conjuntos de palavras-chaves e expressões, as quais classificam os eventos como normais ou anormais.

b) Arquitetura Baseada em *Host* e Distribuída: A solução

proposta baseia-se na análise de registros de *logs* de servidores, caracterizando uma solução baseada em *host*. No entanto, podem existir diversos computadores responsáveis pelo fornecimento de serviços distribuídos pela rede, assim como os componentes para detecção, análise, armazenamento e geração de respostas podem estar distribuídos e trabalhando cooperativamente sem relação hierárquica. Para o registro de *logs* foi aplicado o *Syslog-ng* e para a distribuição dos `logs` projetou-se um modelo aplicando agentes móveis (Seção 5.2.1, Página 50).

c) Período de Ativação: O modelo foi definido para funcionar continuamente. Cada módulo da solução foi implementado deixando essa característica configurável. Dessa forma, pode-se adequar a melhor relação segurança/desempenho para os diferentes ambientes computacionais que venham a aplicar este método.

d) Geração de Respostas: As respostas foram projetadas para agregar a propriedade de pro-atividade ao SDI. Foram definidas duas metodologias, sendo ambas executadas quando o analisador da ferramenta classifica o evento como ataque ou tentativa de ataque a um dos serviços monitorados. A primeira resposta, passiva, é emitida a partir de uma máquina segura e consiste no envio de um *e-mail* alertando administradores sobre a ocorrência de eventos intrusivos. A segunda resposta é classificada como ativa e é implementada por um agente móvel que indisponibiliza o serviço que está sendo atacado. Essas respostas foram projetadas no intuito de proporcionar uma arquitetura de reatividade, a qual pode ser diversificada e ampliada para diferentes classes de ataques e violações. As reações definidas neste trabalho foram implementadas por meio de um modelo baseado em agentes móveis (Seção 5.2.1, Página 50).

5.2 ARQUITETURA DO MODELO COMPUTACIONAL

O modelo arquitetural foi definido para o atendimento aos requisitos desejáveis em SDIs, conforme definição da padronização *CIDF* (geração de eventos, análise, armazenamento e geração de respostas), descrito melhor em Machado (2005). Para isso utilizaram-se alguns componentes tecnológicos e aplicaram-se experiências já conhecidas, tais como análise sequencial de *logs*, avisos administrativos, geração de respostas, entre outras (JUCÁ, 2001).

Além das funções essenciais, criaram-se mecanismos computacionais a fim de implementar requisitos implícitos oriundos dos princípios de segurança de computadores, tais como persistência dos dados, robustez do modelo, disponibilidade e confiabilidade.

A arquitetura completa do SDI pode ser vista melhor na Figura 5, onde está demonstrado os agentes.

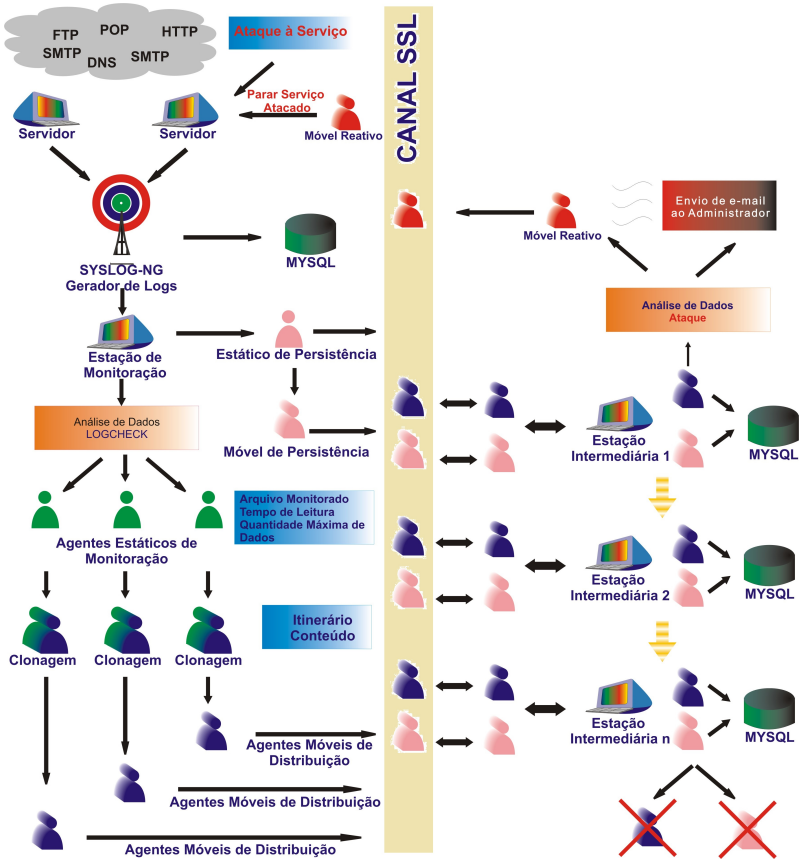

Figura 5 – Arquitetura do Modelo Computacional. (MACHADO, 2005).

Os conceitos dessa arquitetura apresentada na Figura 5 estão melhores descritos nas próximas subseções.

5.2.1 Modelo de Agentes

O modelo de agentes complementa as atividades da ferramenta de detecção de intrusão, sendo responsável pelo armazenamento dos eventos e geração de respostas. A arquitetura de agentes tem por finalidade monitorar os arquivos gerados pelo *Logcheck*, distribuir esses registros de auditoria em um conjunto de servidores confiáveis e prover mecanismos de respostas em situações interpretadas como ataque.

A implementação desses requisitos aplicou o paradigma de agentes móveis, os quais foram escolhidos como a melhor representação para as células do corpo humano. Essa tecnologia foi selecionada em função de importantes propriedades, tais como autonomia, cooperatividade, reatividade, pró-atividade e persistência. Em adicional, as características do problema levam a necessidade da aplicação de propriedades robustas proporcionadas pelos agentes móveis, incluindo a mobilidade, confiabilidade, adaptabilidade e representatividade.

O modelo arquitetural do sistema de agentes é apresentado na Figura 6 e seus componentes descritos a seguir:

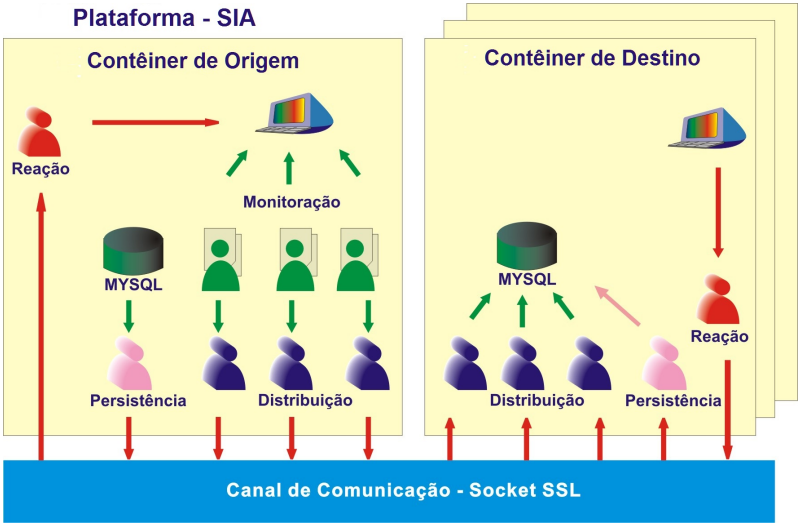

Figura 6 – Arquitetura de Agentes. Adaptado de Machado (2005).

a) **Plataforma SIA**: O termo SIA tem origem na expressão Sistema Imunológico Artificial e foi utilizado como nome da pla-

taforma. Representa o domínio da aplicação, consistindo na disposição física e lógica dos componentes. A plataforma SIA é composta por: Contêiner de Origem, Contêiner de Destino, Serviço de Comunicação e diferentes classes de Agentes.

b) **Contêiner de Origem**: O Contêiner de origem está definido no servidor que está sendo monitorado. Embora, para efeito do modelo e dos experimentos, tenha-se aplicado um único contêiner de origem, pode-se estender para diversos servidores. Esse contêiner é composto pelos agentes estáticos de monitoração que ficam constantemente verificando os arquivos de eventos oriundos do *Logcheck*, agentes móveis de distribuição que armazenam os eventos nos contêineres de destino e por agentes estáticos de persistência que são responsáveis pela robustez do processo de distribuição de eventos.

c) **Contêiner de Destino**: Os contêineres de destino estão localizados em servidores ou estações consideradas seguras. Os agentes móveis de distribuição, provenientes do contêiner de origem, registram os eventos em todos os contêineres de destino, permitindo o armazenamento de informações para análises futuras. A utilização de múltiplos destinos evita a paralisação do sistema em situações de falha de comunicação e constitui uma decisão de projeto que agrega a característica de redundância ao modelo. No primeiro contêiner de destino, os eventos são analisados e são iniciados os processos reativos. Os agentes móveis de persistência passam por todas contêineres de destino armazenando eventos coletados em situações de falha da rede.

d) **Canal de Comunicação**: Mecanismo utilizado para o transporte dos agentes entre distintos contêineres. Os requisitos de segurança, tais como privacidade, confiabilidade, autenticidade e não-repúdio foram alcançados por meio do método de comunicação *socketssl*.

A seguir apresenta-se uma descrição funcional, detalhando os requisitos e características implementadas por cada classe de agentes, o método de mobilidade adotado e o gerenciador de banco de dados aplicado no modelo original. Para uma visão direcionada ao sistema de agentes móveis aplicado deve-se considerar a Figura 5 (Página 49).

AGENTES DE MONITORAÇÃO

Essa categoria é composta por agentes estáticos que possuem como responsabilidade a monitoração dos arquivos gerados pelo *software Logcheck* (*Ataque.sia*, *Violacoes.sia* e *Eventos.sia*). Existe uma instância dessa classe de agentes para cada arquivo monitorado. Ao serem criados, os agentes de monitoração leem um arquivo de configurações com informações importantes para a execução de suas tarefas e de parâmetros que são utilizados pelos agentes de distribuição. O arquivo de configuração contém as seguintes informações:

- **Tipo**: Identifica qual classe de arquivo o agente vai monitorar (ataque, violações de segurança ou eventos de segurança).

- **Arquivo**: Especifica o caminho e o arquivo que o agente será responsável por monitorar.

- **Tamanho**: Especifica a quantidade máxima de dados (em bytes) que será repassada ao agente de distribuição.

- **Tempo**: Define o intervalo de tempo (em *milissegundos*) entre cada acesso ao arquivo monitorado.

- **Email**: Define os endereços eletrônicos para os quais serão enviados os *e-mails* de alerta.

- **Itinerário**: Composto por (N) linhas, representando os contêineres de destino.

Os agentes de monitoração atuam a cada intervalo de tempo definido pelo campo *Tempo* do arquivo de configuração. Sua execução implica na verificação do arquivo monitorado. Caso existam novos eventos, o agente lê o arquivo até seu final ou até atingir o limite de *bytes* definido pelo parâmetro *Tamanho* do arquivo de configuração. Posteriormente, o agente de monitoração cria um agente de distribuição que o parametriza com os registros de *logs* lidos, itinerário a ser percorrido e o conjunto de *e-mails* de administradores. Na Figura 5 (Página 49) esse processo é definido como clonagem.

Os agentes estáticos de monitoração estão representados em verde nas Figuras 5 (Página 49) e 6 (Página 50).

AGENTES DE DISTRIBUIÇÃO

Os agentes de distribuição são responsáveis pelo armazenamento dos *logs* de auditoria nas agências de destino, decodificação dos *logs* e por iniciar processos reativos.

Ao serem criados, os agentes de distribuição recebem do agente de monitoração o conjunto de *logs*, o itinerário a ser percorrido, que são definidos na configuração dos agentes, onde é selecionado os itinerários a partir dos computadores conectados a plataforma principal, e os *e-mails* dos administradores. Como fluxo normal, esses agentes vão até a primeira agência do itinerário (considerado um local seguro), decodificam cada registro de *log* a fim de verificar o tipo de evento, serviço e *host* afetado. Tratando-se de um evento intrusivo (conforme classificação prévia do *Logcheck*), o agente de distribuição envia um relatório, via *e-mail*, aos administradores e instancia um agente móvel reativo (Página 54).

Independente da classificação do *log* e do processo reativo, o agente de distribuição armazena as informações decodificadas na base de dados da primeira agência de destino. Posteriormente os agentes móveis de distribuição passam por todas as agências do itinerário armazenando os *logs* na base de dados local de cada uma dessas agências destinatárias.

Esse método de distribuição permite que o administrador defina um conjunto de computadores seguros para a replicação dos registros auditados e possibilita que qualquer processo reativo seja iniciado a partir desses locais. Essa arquitetura agrega importantes propriedades ao SDI proposto por este modelo, tais como distribuição, robustez e persistência.

Como fluxo alternativo, caso a primeira agência do itinerário não esteja acessível, o agente move-se até a primeira agência do itinerário que esteja disponível, mantendo a ordem definida no arquivo de configuração (campo itinerário). Nessa agência, o agente de distribuição executa as mesmas ações definidas para a primeira agência do itinerário no fluxo normal. Essa abordagem permite generalizar que as atividades principais do agente de distribuição são efetuadas na primeira agência acessível do itinerário.

Como segundo fluxo alternativo, projetou-se uma situação em que o agente não consiga mover-se a nenhuma agência do itinerário. Nesse caso, o agente de distribuição realiza as ações na máquina de origem da mesma forma como se estivesse na primeira agência destino, armazenando os dados em uma base local e eventualmente iniciando

um processo reativo da forma previamente definida. A posterior distribuição desses *logs* é realizada pelos agentes de persistência, que são definidos na Página 54.

Os agentes móveis de distribuição são apresentados em azul nas Figuras 5 (Página 49) e 6 (Página 50).

AGENTES REATIVOS

A estrutura de reatividade proposta neste modelo é baseada em duas estratégias. A primeira, classificada como passiva e mencionada quando da descrição dos agentes de distribuição, consiste no envio de relatórios aos administradores, via *e-mail*, apresentando os registros de *logs* classificados como ataque.

A segunda estratégia, com características proativas, implica na criação de agentes móveis reativos. Esses agentes foram projetados inicialmente para prover a reatividade a ataques direcionados aos quatro (4) serviços monitorados (DNS, FTP, HTTP, POP3 e SMTP) e a ação executada é a finalização do serviço. Esse conceito foi definido como mecanismo de validação do modelo e pode ser estendido para outras reações especializadas.

O ciclo do agente reativo é iniciado com sua criação pelo agente de distribuição. Sua tarefa implica em verificar o serviço e o *host* que estão sofrendo o ataque. Posteriormente o agente move-se até o local atacado e indisponibiliza o serviço afetado. Adicionalmente o agente reativo registra a ação realizada em uma base de dados local e é removido.

Os agentes móveis reativos são apresentados em vermelho nas Figuras 5 (Página 49) e 6 (Página 50).

AGENTES DE PERSISTÊNCIA

Os agentes de persistência foram definidos para garantir a distribuição dos *logs* nas circunstâncias em que o agente de distribuição não consiga fazê-lo. Sua arquitetura consiste em um agente de persistência estático que fica verificando em intervalos regulares de tempo se existem novos dados na base de dados local do servidor monitorado. Caso exista, significa que o agente de distribuição não conseguiu mover-se até as agências de destino. Nessas condições, o agente de distribuição armazena os registros em uma base de dados local.

A partir desse contexto, o agente estático de persistência passa a verificar se agências do itinerário estão acessíveis. No momento em que se tornam disponíveis, o agente estático de persistência cria um agente móvel de persistência que faz a distribuição dos *logs* nas agências de destino da mesma forma que seria realizado pelo agente de distribuição, porém sem iniciar o processo reativo.

Essa classe de agentes realiza a persistência e a distribuição dos registros de auditoria em situações de instabilidade da rede. Com isso contribui para a robustez e tolerância a falhas do modelo.

Os agentes de persistência são apresentados em rosa nas Figuras 5 (Página 49) e 6 (Página 50).

CANAL DE COMUNICAÇÃO SOCKET SSL

O desenvolvimento do protocolo Secure Socket Layer (SSL) foi motivado pela demanda por conexões seguras em aplicações internet. O SSL foi implementado em 1995 pela *Netscape Corporation* tendo como objetivo principal prover uma comunicação com privacidade e confiabilidade entre aplicações.

A arquitetura SSL foi projetada para utilizar a camada de transporte TCP, provendo um serviço seguro e confiável de comunicação fim-a-fim às camadas superiores, em particular ao *HTTP* (STALLINGS, 2003).

O protocolo SSL passou por várias versões e está atualmente na 3.0, a qual admite uma variedade de algoritmos e opções distintas (TANEMBAUM, 2003).

Essas propriedades de segurança motivaram a escolha do mecanismo de comunicação definido como *socketssl*, o qual utiliza SSL, Figuras 5 (Página 49) e 6 (Página 50), e é disponibilizado pelo *JADE* em conjunto com o *Java*. Com isso torna-se possível a mobilidade, comunicação e implementação de agentes estáticos e móveis atendendo aos requisitos de confiabilidade e integridade almejadas em SDIs.

SISTEMA GERENCIADOR DE BANCO DE DADOS MYSQL

Em qualquer tipo de aplicação, o armazenamento de informações em bases de dados constitui uma solução bastante flexível, rápida e segura, além de facilitar a geração de estatísticas e permitir a integração com outras ferramentas de análise. Essas razões motivaram a utilização

de um sistema gerenciador de banco de dados neste modelo, como mecanismo de armazenamento dos eventos de segurança. Conforme apresentado nas Figuras 5 (Página 49) e 6 (Página 50), cada agência de origem e destino possui uma base de dados.

Entre as diferentes alternativas tecnológicas, o *Mysql* foi escolhido por ser uma plataforma de código aberto e por possuir bons índices de performance, confiabilidade, facilidade de utilização, adaptabilidade a diferentes tamanhos de bases de dados, entre outros.

5.3 CONSIDERAÇÕES SOBRE O MODELO COMPUTACIONAL

Por meio da definição de cada componente do modelo computacional apresentaram-se os elementos que constituem o trabalho original (MACHADO, 2005), com algumas alterações para adaptação da plataforma *JADE* referente a nomenclatura utilizada na arquitetura dos agentes, caracterizando um SDI baseado em *Host*, com arquitetura distribuída, reatividade ativa e passiva, executado em tempo contínuo e empregando o método de detecção por anomalia.

Este modelo computacional é resultado da aplicação de princípios e propriedades abstraídas do sistema imunológico humano. E sua principal vantagem diante dos SDIs hoje presentes no mercado, como o *Snort* (CISCO, 2014), é sua detecção por anomalia, diferentemente da detecção por abuso, onde é reconhecido somente um conjunto de regras conhecidas, os quais ataques novos geralmente não são reconhecidos, pois suas assinaturas ainda não estão presentes no conjunto de regras. Outro ponto importante é que pela utilização de agentes móveis, e uma arquitetura distribuída, temos a grande vantagem de enviar o processamento da máquina de origem (que está sendo atacada), para uma máquina remota, segura, onde a partir do momento que o agente móvel saí da máquina de origem para a máquina de destino, por mais que a máquina de origem perca conexão neste tempo, o processamento ainda continua, e os administradores são avisados do ataque, e grande parte do processamento e dados são enviados para uma máquina que não necessita de acesso direto a internet, assim, uma máquina com melhor segurança, mantendo a integridade do sistema de detecção de intrusão.

6 ADAPTAÇÃO DO SISTEMA DE DETECÇÃO DE INTRUSÃO

Após uma revisão do sistema de detecção de intrusão baseado em um sistema imunológico humano e os conceitos correlatos, este capítulo aborda a adaptação da ferramenta para a nova plataforma de agentes móveis, o *JADE* (*Java Agent DEvelopment Framework*), em substituição à plataforma original *Grasshopper*, e as informações relativas a esse processo de alteração do ambiente, e dos agentes estáticos e móveis do sistema de detecção de intrusão.

6.1 ESCOLHA DA PLATAFORMA

Como a plataforma de agentes móveis do trabalho original, o *Grasshopper*, tornou-se obsoleta, não se encontrando mais os arquivos para sua instalação e configuração, é de extrema importância para o funcionamento do SDI a troca de plataforma. Com isto, foi feito um estudo sobre as plataformas existentes, o qual se utilizou os trabalhos de Gupta e Kansal (2011), Nguyen et al. (2002), Trillo, Ilarri e Mena (2007) e Rajguru e Deshmukh (2011), onde é feito um estudo comparativo de plataforma de agentes móveis, para a escolha da plataforma a ser utilizada neste trabalho. Entre as plataformas presentes nestes comparativos, as principais desenvolvidas em *Java* são: *Aglets*, *JADE*, *Voyager*, *Grasshopper* e *SPRINGS*. Os autores apresentam a plataforma *SPRINGS* como uma plataforma confiável por fornecer escalabilidade e confiabilidade com um grande número de agentes móveis. E o *JADE* como plataforma mais equilibrada, por ser totalmente desenvolvida em *Java*, suportar diferentes tipos de dispositivos e proporcionar um forte mecanismo de segurança. Além disso, dentre as plataformas testadas, ainda temos as plataformas *Aglets* e *Voyager*, que não utilizam o padrão FIPA. Com isto, os principais pontos para a escolha da plataforma foram para as plataformas desenvolvidas em *Java* e que seguem o padrão FIPA, e entre as duas plataformas que seguem este princípio, o JADE e o SPRINGS, foi dado preferência ao *JADE* por seus recursos como segurança e interface gráfica, os quais não existem ou são limitados no *SPRINGS*, e por haver grande aceitação pela comunidade de agentes móveis. A Tabela 1 mostra as plataformas e os principais pontos abordados. E nas próximas subseções é apresentado as principais características da plataforma antiga e da plataforma nova,

com um resumo sobre *Grasshopper* e um estudo sobre o *JADE*.

Tabela 1 – Comparação de Agentes Móveis baseados em *Java*. Adaptado de (GUPTA; KANSAL, 2011).

Aspecto	**Aglets**	***JADE***	**Voyager**	**Grasshop.**	**SPRINGS**
Segurança	Limitado	Forte	Limitado	Limitado	Limitado
Comunicação	Síncrono, Assíncrono	Assíncrono	Todos os métodos	Síncrono, Assíncrono	Síncrono, Assíncrono
Mobilidade	Protocolo de transferência Aglet	Serviço de Mobilidade de Agente Embutido	Série de Objeto Java	*Proxies* Dinâmicas	*Proxies* Dinâmicas
Ferramentas Gráficas	Alguma	Sim	Não	Sim	Não
Padronização	MASIF	FIPA	Não	FIPA, MASIF	Não
Organização	IBM Tokyo Research	Telecom Italia Lab	Object Space	IKV++	Distributed Information Systems Group

6.1.1 A Plataforma *Grasshopper*

A plataforma *Grasshopper*, a qual foi utilizada no trabalho original, foi desenvolvida pela IKV++, uma empresa alemã de tecnologia para comunicação e serviço de informação, em 1999. Sua última versão é datada de 2004, e a plataforma está de acordo com o padrão MASIF e FIPA. O sistema é composto de regiões e agências, no qual as agências são responsáveis por controlar a execução, gerenciamento, transporte, comunicação e outras funções dos agentes.

O *Grasshopper* tem uma interface gráfica (GUI) para gerenciar os agentes, agências e regiões, e é desenvolvido em *Java*, e de acordo com o desenvolvedor, teoricamente deve rodar em qualquer sistema que tenha *Java Runtime Enviroment* (*JRE*), mas para seu funcionamento é necessário a instalação do executável (*Grasshopper.exe*) para *Windows* e a execução do *shell script* (*Grasshopper.sh*) nos sistemas *Unix*, e a sua correta configuração.

A comunicação entre agentes é essencial para o funcionamento de sistemas multi-agentes. Na plataforma *Grasshopper* a comunicação se dá através de *proxy* dinâmico, por meio de mensagens síncronas, onde o servidor de uma região atualiza todo o *proxy* antes de usá-lo, e com isto, podendo existir algum gargalo entre as comunicações desta plataforma. Outro ponto negativo da comunicação do *Grasshopper* é que

uma chamada a um agente em movimento pode ser recebida e executada por sua cópia ainda no computador de origem, que será removida logo que o agente chegue no destino. E quanto a sua segurança, o *Grasshopper* oferece tanto segurança interna com autorização dos agentes, como externa, utilizando SSL e certificados X.509.

6.1.2 A Plataforma *JADE*

Com o surgimento da empresa Telecom Italia em 1994, uma plataforma mais moderna foi idealizada e construída, sendo denominada, plataforma *JADE*, que é um *framework* totalmente implementado em *Java*, e que segue as especificações do padrão FIPA. A plataforma foi desenvolvida pelo TILAB (Telecom Italia Lab) em julho de 1998, e é um software livre distribuído pela Telecom Italia, sobre os termos da licença LGPL (Lesser General Public License Version 2), sendo sua última versão datada de 28 de março de 2014.

Em maio de 2003, foi criado o *JADE* Board, ou seja, um grupo de empresas que supervisiona e gerencia o projeto *JADE*. Entre elas temos a Telecom Italia, Motorola, Whitestein Technologies AG, Profactor GmbH e a France Telecom R&D.

O sistema, basicamente, funciona sobre contêineres: o contêiner principal e os contêineres remotos, onde os agentes "vivem". Assim como o *Grasshopper*, *JADE* também tem uma interface gráfica, onde é possível ver e gerenciar esses contêineres e agentes, configurar os *logs* e enviar mensagens para os agentes. Por ser uma plataforma desenvolvida em *Java*, funciona em qualquer sistema com o *Java Runtime Enviroment* (*JRE*), não precisando de instalação, pois o *JADE* funciona a partir de uma biblioteca *Java*, de extensão *Java Archive* (*.jar*), onde só precisa adicioná-la ao *classpath* do projeto e fazer as devidas importações.

A comunicação na plataforma *JADE*, onde não existe o conceito de *proxy* utilizado pelo *Grasshopper*, se dá através de mensagens assíncronas, onde o agente de origem consulta o Serviço de Gerenciamento de Agentes (Agent Management Service - AMS) sobre o agente de destino de acordo com a especificação FIPA, dando ao *JADE* um melhor gerenciamento das mensagens do sistema, ou seja, não dependendo temporalmente de outros agentes para fazer a comunicação, assim, o agente receptor da mensagem não precisa ter conexão direta com o agente transmissor, garantindo a comunicação entre os agentes mesmo em conexões oscilantes. Junto disto, o agente tem uma *"thread"* de

mensagens, as quais a medida que são recebidas, vão para esta lista, onde o agente pode gerenciá-la de acordo com suas prioridades.

Em termos de segurança, *JADE* propicia os requisitos básicos, tais como, autenticação (do usuário), autorização (sobre as ações realizadas pelos agentes), integridade e não-repúdio são dadas pelas assinaturas das mensagens e confidencialidade (criptografia). O *JADE* foi construído contendo um *plugin*, que pode ser configurado, para usar quatro serviços de *kernel*, entre eles, o serviço de segurança que faz autenticação, o serviço de permissão que faz autorização, serviço de assinatura que assina mensagens e o serviço de criptografia, que criptografa as mensagens.

O *JADE* se baseia nos seguintes princípios (TEIXEIRA:2010):

- **Interoperabilidade**: está em conformidade com as especificações FIPA, possibilitando que os agentes possam se comunicar com outros que não estejam executando em ambiente de execução JADE.

- **Uniformidade e Portabilidade**: provê aplicações com um conjunto homogêneo de *APIs* (Application Programming Interfaces) que são independentes da estrutura subjacente e da versão Java (o ambiente de execução JADE provê as mesmas *APIs* tanto para *J2EE* como para *J2SE* e *J2ME* e, em teoria, o desenvolvedor pode escolher o ambiente de execução *Java* em tempo de compilação.

- **Facilidade de Uso**: a complexidade do *middleware* é ocultada por trás de um simples e intuitivo conjunto de *APIs*.

- **Filosofia *Pay-as-you-go***: programadores não são obrigados a usar todas as características providas pelo *middleware*. Aquelas que não serão utilizadas, não requerem nenhum tipo de conhecimento a respeito por parte do programador, tampouco adiciona qualquer sobrecarga computacional.

O *JADE* ainda é uma plataforma que aceita uma vasta classe de dispositivos, incluindo aparelhos celulares, onde os quais somente necessitam de suporte ao *Java MIDP* (*Mobile Information Device Profile*). E conta com o serviço de *publish-subscribe*, onde cada agente pode se inscrever seus serviços em uma lista de "páginas-amarela", e assim, outros agentes conseguem encontrá-lo de acordo com seus serviços, não havendo necessidade de se conhecer todos os agentes na plataforma. O *JADE* ainda aceita integração com outras tecnologias, como *applets* e *web services*, e tem o *kernel* extensível, permitindo aos programadores adicionarem funcionalidades a aplicação (TEIXEIRA, 2010). A

partir deste breve resumo sobre o *JADE*, é mostrado a arquitetura da plataforma.

6.1.2.1 Arquitetura da Plataforma *JADE*

Os principais elementos do *JADE* são: plataforma, contêineres e agentes. Onde os agentes habitam os contêineres, e estes, contêineres, juntos formam a plataforma. Esse conceito da plataforma *JADE* pode ser melhor visto na Figura 7.

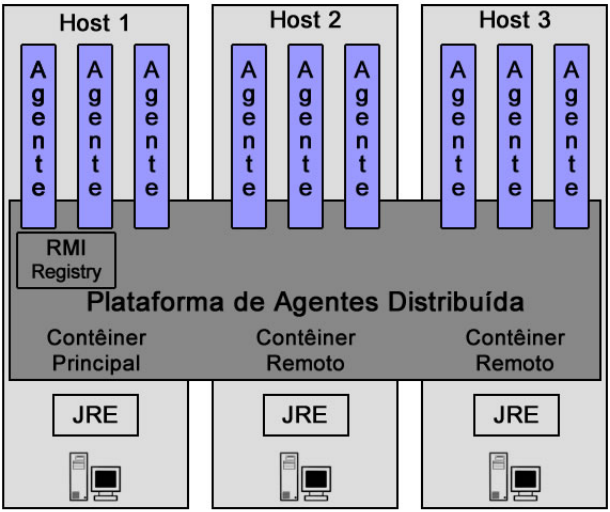

Figura 7 – Arquitetura da plataforma *JADE*. Adaptado de (TEIXEIRA, 2010)

A plataforma necessariamente tem um contêiner principal, o qual é responsável pelo gerenciamento dos agentes, dos contêineres, e da plataforma. Com isto, as principais funções do contêiner principal são (TEIXEIRA, 2010):

- Administrar o *Container Table* (CT), que é o registrador das referências de objeto e endereços de transporte de todos os outros nós contêineres que compõem a plataforma;
- Gerenciar a *Global Agent Description Table (GADT), que é o registrador de todos os agentes presentes na plataforma (incluindo*

seus estados correntes e suas localizações);

- Hospedar os três agentes principais definidos no modelo de referência FIPA;

Assim, habitando o contêiner principal, temos os agentes AMS, DF e ACC, que são automaticamente instanciados e inicializados pelo *JADE* na criação do contêiner, onde o AMS é responsável pela plataforma, sendo necessário todo e qualquer agente se registrar ao AMS, obtendo assim um *AID* válido, ou seja, um identificador único dentro da plataforma. Já o DF, é responsável pelas "páginas-amarelas", sendo utilizado por qualquer agente que necessite subscrever seus serviços e buscar serviços ofertados por outros agentes. E o ACC é responsável pela comunicação entre os agentes de dentro e fora da plataforma, através da comunicação padrão FIPA.

Por sua Importância, e por ser um ponto importante que pode falhar, a plataforma ainda permite replicar o contêiner principal, assim garantindo a funcionalidade da plataforma nos casos de falha, permitindo ao administrador controlar o nível de tolerância a falha, o nível de escalabilidade e o nível de distribuição da plataforma.

6.2 SUBSTITUIÇÃO DA PLATAFORMA

Para a substituição da plataforma, foi feito um estudo sobre o *Grasshopper* e sobre o *JADE*, onde foi verificado suas diferenças nos conceitos, na estrutura e na implementação. Com isto, o primeiro ponto a ser verificado, é a criação da plataforma, que anteriormente necessitava da instalação e configuração da plataforma *Grasshopper*, e agora, com o *JADE*, foi feito a iniciação da plataforma através do código, o qual é obtido um *JADE Runtime* e a partir dele, criado um contêiner, tudo isto em tempo de execução. Com isto, a biblioteca *JADE* é mantida junto ao código do SDI, e não é necessário a instalação ou configuração adicional ao SDI.

O passo seguinte é a transformação das regiões do *Grasshopper*, onde todo o código referente a isto, teve que ser revisto e transformado em Contêineres do *JADE*, e onde era instanciado um *GrasshopperAddress* (Endereço do *Grasshopper*), foi necessário fazer algumas modificações para transformá-lo em Contêiner, através da utilização dos nomes de contêineres, ao invés de endereços. Ou seja, todos os objetos que antes eram tratados como endereços do *Grasshopper*, foram substituídos por objetos de texto (*Strings*), contendo o nome do

contêiner, e com este nome, era instanciado um contêiner propriamente dito quando necessário.

Posteriormente foi feita a mudança nos agentes, onde as principais mudanças são do método *init* do *Grasshopper*, o qual era utilizado para iniciar as variáveis e recebia algumas destas através de argumentos da assinatura do método, onde, no *JADE*, o método para iniciação de variável é o *setup*, que não recebe variáveis pela assinatura, e sim pela execução de outro método, o *getArguments* que retorna uma lista de objetos genéricos que foram passados por outro objeto na criação deste agente. Ainda na mudança dos agentes, a maior delas foi no comportamento, onde o *Grasshopper* utilizava o método *live* para definir o que o agente executava em vida. No *JADE* é permitido ao agente ter um ou mais comportamentos, onde cada comportamento é uma classe diferente descrevendo a execução daquele agente, o qual podem ser alterados em tempo de execução através do método *addBehaviour*. Como boas práticas, quando um comportamento é utilizado somente por um agente, a classe deste comportamento é criada como classe interna dentro do próprio agente.

Estas foram as principais mudanças na substituição da plataforma, sendo necessárias para o correto funcionamento do SDI novamente. Outras mudanças foram na adição de um agente que se inscreve junto ao *AMS* e fica ouvindo sobre as adições e remoções dos contêineres, assim o agente consegue manter uma lista atualizada dos contêineres conectados a plataforma. Portanto, o SDI consegue mostrar uma lista de contêineres possíveis para serem escolhidos como lista de itinerários.

E por último, a adição de autenticação entre os contêineres, os quais foram criados uma *Trust Store Java*, uma *Keystore Java* e, a partir dessa *Keystore*, um certificado para cada contêiner da plataforma. A *Keystore* foi criada utilizando o algoritmo de chave privada do tipo *DSA*. Com isto, foi adicionado o certificado de cada contêiner, contendo as chaves públicas, às *Truststore* dos outros contêineres, para que assim, somente os contêineres autenticados possam se conectar na plataforma.

6.3 CONSIDERAÇÕES FINAIS

O *JADE* frente ao *Grasshopper* é uma plataforma mais atual e melhor aceita pela comunidade em geral, sem contar que o *Grasshopper* foi descontinuado, não encontrando mais os arquivos necessários

para sua utilização. Através deste capítulo, foi constatado que provavelmente o SDI ganharia uma melhor velocidade e confiabilidade na troca de plataforma, do *JADE* para o *Grasshopper*, o qual é ratificado no próximo capítulo. Assim, depois do estudo realizado e verificado as mudanças necessárias, foi feito a troca de plataforma, respeitando as referências de cada plataforma, como Região e Agência do *Grasshopper*, para Plataforma e Contêiner do *JADE*.Com isto, para este trabalho e para constatação dos testes, foi abstraído, por não se encontrar necessário, as ferramentas auxiliares do SDI, que não foram desenvolvidas no trabalho original, como o *Syslog-ng*, *Logcheck* e *MySQL*. Portanto, no próximo capítulo é demonstrado os resultados dos testes de desempenho da nova plataforma comparado a plataforma antiga, e as melhorias adicionadas a ferramenta, sem contar na possibilidade de funcionamento de uma ferramenta importante do laboratório que até então se encontrava parada.

7 DISCUSSÃO DOS RESULTADOS

Depois da translação da plataforma antiga para a nova, foi constatado seu correto funcionamento através de testes, utilizando os mesmos *logs* de resultados do trabalho original, os quais foram gerados através do *Syslog-ng* e *Logcheck*, ou seja, através de *logs* reais. Com isto, neste capítulo são apresentados os resultados dos testes de desempenho e a comparação com a plataforma antiga, através de agentes desenvolvidos junto ao trabalho original, os quais também foram adaptados para a nova plataforma para que fosse possível a realização destes.

7.1 AVALIAÇÃO DE DESEMPENHO DE AGENTES MÓVEIS

Esta avaliação consistiu em determinar o desempenho da mobilidade de agentes móveis *JADE* com tamanhos distintos em redes *Ethernet* 100 Mbps e 1000 Mbps. Esses experimentos foram realizados em ambiente controlado aplicando os métodos de transmissão *Socket* e *Socketssl*.

Os detalhes relativos aos métodos experimentais, análise estatística aplicada, classes de resultados obtidos e sua discussão são apresentados nas próximas subseções.

7.1.1 Descrição do Método Experimental de Avaliação de Desempenho da Mobilidade

O desempenho dos agentes foi avaliado definindo-se um ambiente controlado, de forma que o único tráfego da rede foi causado pela mobilidade dos agentes.

Avaliou-se o tempo de migração dos agentes com segmentos de dados no intervalo de 0 a 2 MB, variando de 100 em 100 KB (0 KB, 100 KB, 200 KB, 300 KB, 400 KB, 500 KB, 600 KB, 700 KB, 800 KB, 900 KB, 1000 KB, 1100 KB, 1200 KB, 1300 KB, 1400 KB, 1500 KB, 1600 KB, 1700 KB, 1800 KB, 1900 KB e 2000 KB). Coletaram-se mil (1000) amostras de tempo, por segmento, nas duas tecnologias de redes (Ethernet 100 Mbps e 1000 Mbps). Esses experimentos foram realizados para os métodos de transmissão *socket* e *socketssl*.

Para isso utilizou-se um ambiente computacional composto de dois computadores, sendo o computador referente ao contêiner princi-

pal:

- Processador Intel Core i7 740QM 1.73 GHz, 8 GB de memória DDR3, placa de rede 10/100/1000 Mbps.
- Sistema operacional *Microsoft Windows 8.1*.
- Máquina virtual *Java JDK 8u5*.

E o computador referente ao contêiner remoto com a seguinte configuração:

- Processador Intel Core i3 330M 2.13 GHz, 3 GB de memória DDR3, placa de rede 10/100/1000 Mbps.
- Sistema operacional *Linux Ubuntu 14.04LTS*.
- Máquina virtual *Java JRE 8u5*.

Para a conexão dos dois computadores, foi utilizado um cabo de rede ligado diretamente entre as placas de redes, para que somente os agentes trafegassem pela rede, sem sofrer com ruídos externos, e foi utilizado a plataforma *JADE* versão 4.3.2.

7.1.2 Desempenho dos Agentes Aplicando o Método Socket

A primeira categoria de experimentos foi realizada aplicando o método de transmissão de agentes *socket* e variando-se o tamanho dos segmentos de dados dos agentes e as tecnologias das redes, conforme descrito na Seção 7.1.1 (Página 65). Na Tabela 2 são apresentadas as médias e desvio padrão dos tempos gastos, em segundos, para a mobilidade dos agentes para os distintos tamanhos de agentes e tecnologias de redes. Na Figura 8 é apresentado um gráfico demonstrando a diferença de desempenho dos diferentes tamanhos de agentes nas distintas tecnologias de redes.

Entre diversas análises possíveis, a curva demonstra que o desempenho dos agentes em rede de 100Mbps é linear. Observações preliminares permitem conceber as hipóteses que os agentes apresentaram melhores desempenho em redes Ethernet 1000 Mbps comparados a redes Ethernet 100 Mbps, principalmente para tamanho de agentes maiores.

Tabela 2 – Desempenho de Agentes Móveis com Método de Mobilidade Socket

	100 Mbps		*1000 Mbps*	
Tamanho	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**
0 KB	0,0036035	0,005271368	0,00370350	0,006068127
100 KB	0,0123995	0,005014309	0,00818000	0,021673378
200 KB	0,0217650	0,006227162	0,00918800	0,016473085
300 KB	0,0308360	0,002762560	0,01299200	0,023194459
400 KB	0,0403525	0,003355705	0,01689250	0,027748823
500 KB	0,0496095	0,006420474	0,01986750	0,028008073
600 KB	0,0587030	0,004671897	0,02136700	0,026335373
700 KB	0,0677900	0,004429732	0,02621850	0,032964453
800 KB	0,0770035	0,006231297	0,02881250	0,023785202
900 KB	0,0863450	0,003249236	0,03254900	0,028850389
1000 KB	0,0952825	0,004195420	0,03700800	0,040241071
1100 KB	0,1043670	0,006713560	0,03825800	0,039201552
1200 KB	0,1132025	0,004589587	0,04288250	0,041868472
1300 KB	0,1222100	0,006816037	0,04459400	0,038038969
1400 KB	0,1309140	0,007779339	0,04561050	0,026722090
1500 KB	0,1399225	0,007665573	0,04674250	0,037780144
1600 KB	0,1506105	0,007359395	0,05275050	0,041825948
1700 KB	0,1600165	0,006099277	0,05505400	0,035037933
1800 KB	0,1688825	0,005721554	0,05885150	0,031155061
1900 KB	0,1785630	0,008366363	0,06045300	0,035060705
2000 KB	0,1879065	0,009591116	0,06064800	0,039493973

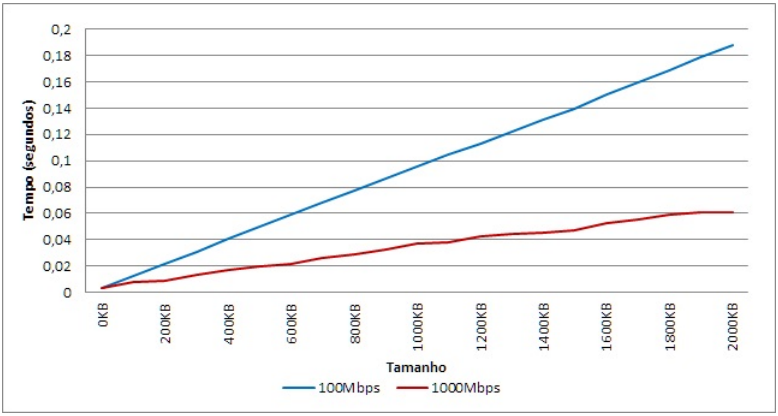

Figura 8 – Desempenho de Agentes Móveis com Método de Mobilidade Socket

7.1.3 Desempenho dos Agentes Aplicando o Método Socketssl

A segunda classe de experimentos foi realizada aplicando o método de transmissão de agentes *socketssl*, protocolo seguro que aplica técnicas

criptográficas, variando-se o tamanho dos segmentos de dados dos agentes e as tecnologias de rede, conforme descrito na Seção 7.1.1 (Página 65). Na Tabela 3 são apresentadas as médias e desvios padrões dos tempos, em segundos, gastos para a mobilidade dos agentes com os distintos tamanhos de segmentos de dados e tecnologias de redes. Na Figura 9 apresenta-se um gráfico demonstrando a diferença de desempenho dos diferentes tamanhos de agentes nas distintas tecnologias de redes.

Tabela 3 – Desempenho de Agentes Móveis com Método de Mobilidade Socketssl

	100 Mbps		*1000 Mbps*	
Tamanho	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**
0 KB	0,77307000	0,016569736	0,77223150	0,017859679
100 KB	0,84861600	0,037679235	0,84030400	0,022347080
200 KB	0,91876750	0,034664500	0,91549650	0,034794130
300 KB	1,00681250	0,038053490	0,98929650	0,037367950
400 KB	1,11380200	0,048173090	1,08662400	0,035006400
500 KB	1,25817150	0,050410957	1,22778050	0,030969375
600 KB	1,46378800	0,129904529	1,38093300	0,023859994
700 KB	1,62498250	0,195661714	1,51669600	0,010075594
800 KB	1,71990000	0,117230331	1,66889350	0,026535090
900 KB	1,85901950	0,126207933	1,80522400	0,029224800
1000 KB	2,04541200	0,127389781	1,94954600	0,038958621
1100 KB	2,18044600	0,021816268	2,12658200	0,025015486
1200 KB	2,41333150	0,006435081	2,35025150	0,011728899
1300 KB	2,66381050	0,117076713	2,60374300	0,017570748
1400 KB	2,78213650	0,006237911	2,73191300	0,058404396
1500 KB	2,91152450	0,004480036	2,85021600	0,007920232
1600 KB	3,03489600	0,004949918	2,97335400	0,010723020
1700 KB	3,16503250	0,004805079	3,09812500	0,015129021
1800 KB	3,25116900	0,022028293	3,22361550	0,014694432
1900 KB	3,42440050	0,005914036	3,34417450	0,015207648
2000 KB	3,43360300	0,009397058	3,35412700	0,016708996

Conforme ilustrado na Figura 9, pode-se observar um comportamento similar ao apresentado nos experimentos aplicando canal de comunicação *socket* (Figura 8). Novamente é observado uma linearidade da curva da rede Ethernet 100 Mbps, e a não-linearidade, na medida que aumenta-se o tamanho do segmento de dados dos agentes da rede Ethernet 1000 Mbps. Neste caso, é notado um comportamento parecido, comparando-se o desempenho das redes Ethernet 100 Mbps e 1000 Mbps do *socket* não criptografado.

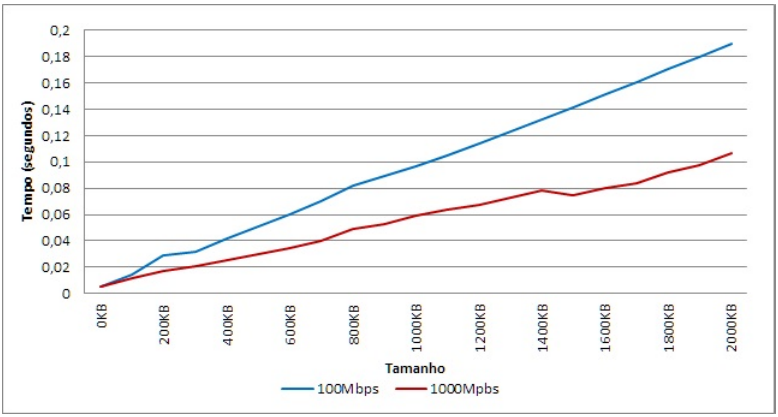

Figura 9 – Desempenho de Agentes Móveis com Método de Mobilidade Socketssl

7.1.4 Desempenho dos Agentes em Redes Ethernet 100 Mbps

Nessa subseção compara-se o desempenho dos agentes móveis em redes Ethernet 100 Mbps variando-se o tamanho dos segmentos de dados dos agentes e o método de mobilidade (*socket* e *socketssl*).

As médias de tempo para a transmissão dos agentes, em segundos, e desvio padrão podem ser observadas na coluna (100 Mbps) da Tabela 2 (Página 67) para canais *socket* e na coluna (100 Mbps) da Tabela 3 (Página 68) para canais *socketssl*.

Na Figura 10 apresenta-se graficamente essa variação de desempenho dos agentes com tamanhos e métodos de mobilidade distintos em redes 100 Mbps, assim, permitindo observar visualmente a diferença de desempenho dos agentes comparando-se os dois métodos de mobilidade, o qual é quase nula, demonstrando que, para redes Ethernet 100 Mbps a velocidade da rede funciona como um gargalo, não havendo diferença na performance entre os agentes criptografados e os não criptografados.

7.1.5 Desempenho dos Agentes em Redes Ethernet 1000 Mbps

Nessa subseção apresenta-se um comparativo de desempenho dos agentes móveis em redes Ethernet 1000 Mbps, variando-se o tamanho dos segmentos de dados dos agentes e o método de mobilidade (*socket* e *socketssl*).

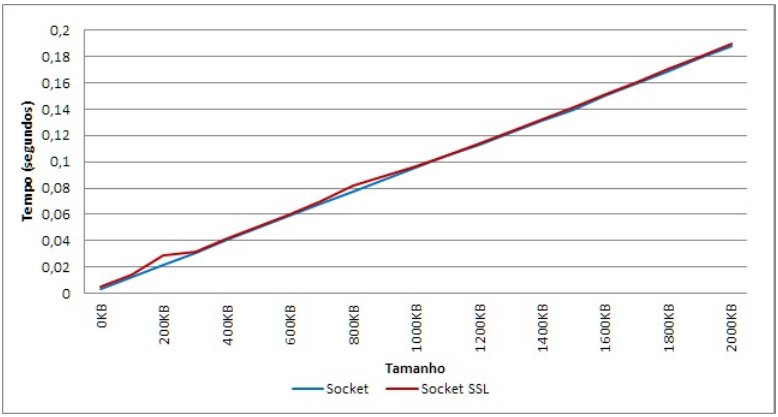

Figura 10 – Desempenho de Agentes em Redes Ethernet 100 Mbps utilizando *Socket* e *Socketssl*

As médias de tempo para a transmissão dos agentes, em segundos, e desvio padrão podem ser observadas na coluna (1000 Mbps) da Tabela 2 (Página 67) para canais *socket* e na coluna (1000 Mbps) da Tabela 3 (Página 68) para canais *socketssl*. Na Figura 11 apresenta-se graficamente essa variação de desempenho de agentes com tamanhos de segmentos de dados e método de mobilidade distintos em redes 1000 Mbps.

As diferenças de desempenho, aplicando-se os dois métodos de mobilidade na rede Ethernet 1000 Mpbs, podem ser observada na Figura 11 e diferentemente da rede Ethernet 100 Mbps, mostrou uma maior diferença de velocidade, principalmente para agentes de maiores tamanhos.

7.1.6 Diferença de desempenho na plataforma *Grasshopper* e *JADE*

Nesta seção será mostrada a diferença entre as plataformas *Grasshopper*, plataforma do trabalho original, e *JADE*, no quesito desempenho, comparando os resultados das redes Ethernet 100 Mbps e 1000 Mbps, sobre *socket* e *socketssl*. Para esta seção do trabalho foi usado os resultados do trabalho original, que por usar uma plataforma em desuso, não foi possível replicar os resultados no mesmo ambiente que os testes da plataforma *JADE*. No trabalho original foi usado o seguinte

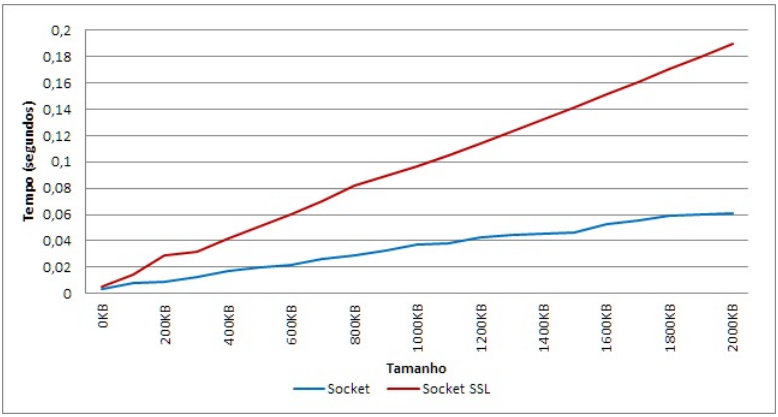

Figura 11 – Desempenho de Agentes em Redes Ethernet 1000 Mbps utilizando *Socket* e *Socketssl*

ambiente computacional:

- Dois computadores Pentium IV 2.4 GHz, 512 MB de memória DDR, placa de rede 10/100/1000 Mbps.
- Ligações ponto a ponto, permitindo que o único tráfego na rede seja a transmissão dos agentes.
- Sistema operacional *Linux*, *kernel* 2.4.3, distribuição *Suse* 9.0.
- Máquina virtual *Java JDK 1.4.2*.
- Plataforma de Agentes Móveis *Grassshopper 2.2.4*.

Com isto, na Tabela 4 temos os resultados dos experimentos para as redes Ethernet 100 Mbps e 1000 Mbps sobre *socket*, e na Tabela 5, as mesmas velocidades de rede sobre *socketssl*.

A partir destes resultados da Tabela 4 e 5, comparando-os com os resultados obtidos nas subseções anteriores, temos as Figuras 12, 13, 14 e 15 comparando os respectivos resultados das redes Ethernet 100 Mbps e 1000 Mbps sobre *socket* e Ethernet 100 Mbps e 1000 Mbps sobre *socketssl* na plataforma *JADE* e na plataforma *Grasshopper*.

A partir das Figuras 12, 13, 14 e 15 é possível observar que o *JADE* tem um melhor desempenho sobre o *Grasshopper*, independente da velocidade da rede ou do uso da criptografia.

Tabela 4 – Desempenho de Agentes Móveis da plataforma *Grasshopper* com Método de Mobilidade *Socket*

	100 Mbps		*1000 Mbps*	
Tamanho	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**
0 KB	1,18416200	0,02942461	1,17559250	0,03371242
100 KB	1,36350400	0,04151405	1,30148500	0,03325058
200 KB	1,56140900	0,04700810	1,45443700	0,04098876
300 KB	1,76146400	0,05740669	1,58667750	0,04654189
400 KB	1,95876650	0,06678727	1,73310900	0,04598066
500 KB	2,15682750	0,07240694	1,86184950	0,04308284
600 KB	2,35261450	0,08022143	2,00713450	0,04394423
700 KB	2,60349800	0,07449215	2,18077750	0,02615715
800 KB	2,77139800	0,06733055	2,35123050	0,01743804
900 KB	2,96253200	0,05571130	2,47458200	0,07055979
1000 KB	3,17182650	0,06178013	2,58389550	0,04277578
1100 KB	3,41176000	0,08382144	2,80313400	0,02037834
1200 KB	3,69280400	0,05734406	3,04428350	0,02088805
1300 KB	4,15431700	0,04206369	3,34762150	0,02236789
1400 KB	4,40062300	0,04340547	3,52490700	0,02038099
1500 KB	4,59327700	0,04323936	3,65802950	0,02449934
1600 KB	4,73695000	0,05590412	3,78388900	0,02028918
1700 KB	4,90438050	0,04775342	3,84206950	0,01933812
1800 KB	5,09105400	0,04767082	3,91596650	0,01873305
1900 KB	5,28920900	0,06837095	4,04282600	0,02656558
2000 KB	5,34069900	0,06293536	4,12012450	0,03807015

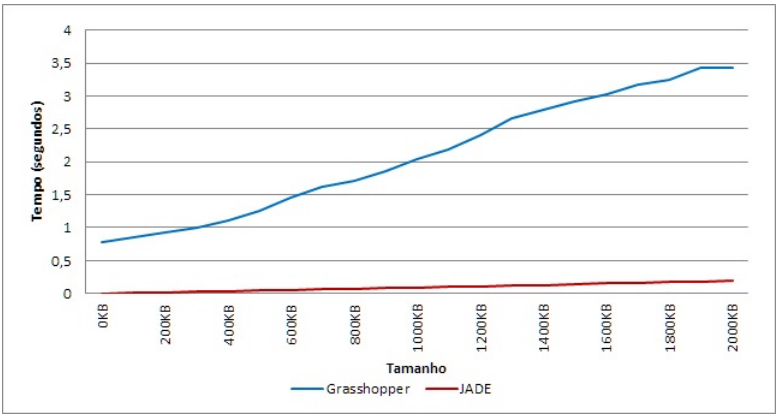

Figura 12 – Desempenho de Agentes em Redes Ethernet 100 Mbps utilizando *Socket* nas plataformas *JADE* e *Grasshopper*

7.1.7 Análise dos Resultados

Essa subseção é destinada à discussão dos resultados obtidos nos experimentos e decisões de projeto resultantes dessa análise, que foram

Tabela 5 – Desempenho de Agentes Móveis da plataforma *Grasshopper* com Método de Mobilidade *Socketssl*

	100 Mbps		*1000 Mbps*	
Tamanho	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**	**Tempo Médio (Segundos)**	**Desvio Padrão (Segundos)**
0 KB	1,18929600	0,03419341	1,17559250	0,03371242
100 KB	1,32375850	0,03991057	1,30148500	0,03325058
200 KB	1,46853900	0,03868240	1,45443700	0,04098876
300 KB	1,62689650	0,04884414	1,58667750	0,04654189
400 KB	1,80914150	0,08666050	1,73310900	0,04598066
500 KB	1,95006000	0,11191249	1,86184950	0,04308284
600 KB	2,09101000	0,11448758	2,00713450	0,04394423
700 KB	2,26637750	0,10824124	2,18077750	0,02615715
800 KB	2,36339800	0,08136181	2,35123050	0,01743804
900 KB	2,51008450	0,04089204	2,47458200	0,07055979
1000 KB	2,63121350	0,05148598	2,58389550	0,04277578
1100 KB	2,82487550	0,04264686	2,80313400	0,02037834
1200 KB	3,07263750	0,01887154	3,04428350	0,02088805
1300 KB	3,38237600	0,01917658	3,34762150	0,02236789
1400 KB	3,56935600	0,03833343	3,52490700	0,02038099
1500 KB	3,70139150	0,02022085	3,65802950	0,02449934
1600 KB	3,83057400	0,02041098	3,78388900	0,02028918
1700 KB	3,89418000	0,01986919	3,84206950	0,01933812
1800 KB	3,96888000	0,01926772	3,91596650	0,01873305
1900 KB	4,09919100	0,01931410	4,04282600	0,02656558
2000 KB	4,12418000	0,02029706	4,12012450	0,03807015

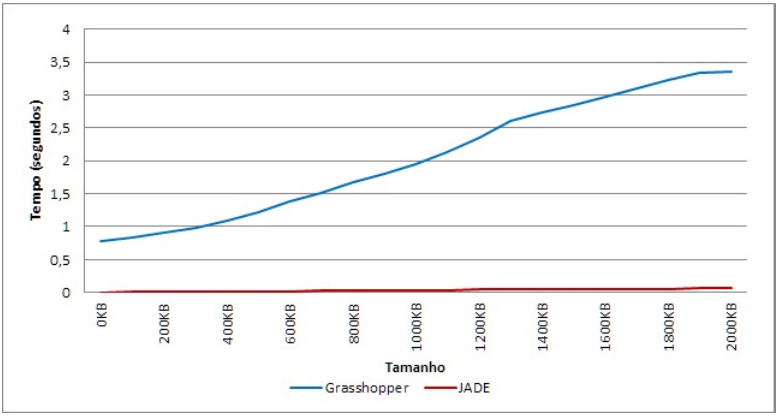

Figura 13 – Desempenho de Agentes em Redes Ethernet 1000 Mbps utilizando *Socket* nas plataformas *JADE* e *Grasshopper*

aplicadas ao SDI proposto neste trabalho.

Os resultados estatísticos evidenciaram que os agentes com dis-

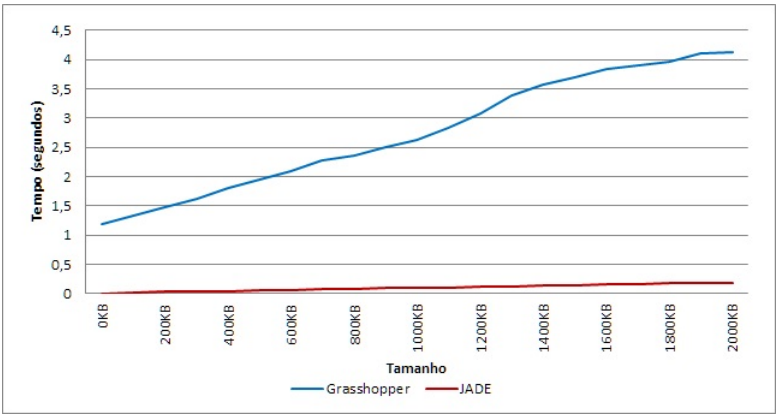

Figura 14 – Desempenho de Agentes em Redes Ethernet 100 Mbps utilizando *Socketssl* nas plataformas *JADE* e *Grasshopper*

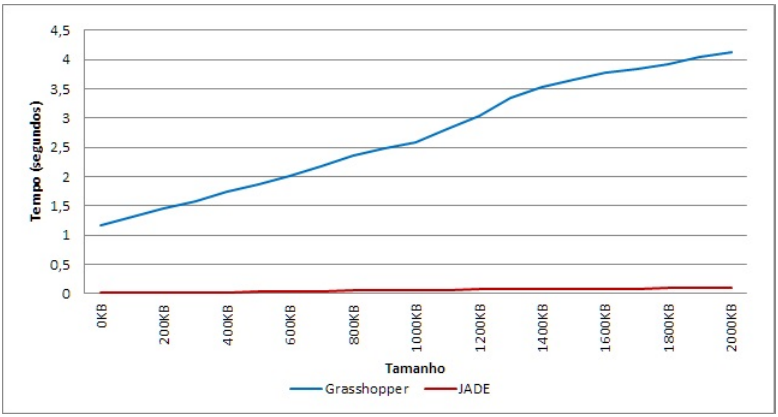

Figura 15 – Desempenho de Agentes em Redes *Ethernet* 1000 Mbps utilizando *Socketssl* nas plataformas *JADE* e *Grasshopper*

tintos tamanhos de segmentos de dados apresentaram rendimentos idênticos em redes *Ethernet* 100 Mbps em *socket* e *socketssl*, mostrando que os agentes estão usando a máxima velocidade nesta faixa de rede, diferentemente das redes *Ethernet* 1000 Mbps, onde os agentes sem criptografia (*socket*) tiveram melhores resultados do que os com criptografia (*socketssl*), onde o protocolo de criptografia conta negativamente no desempenho dos agentes.

Diante dos resultados e dessas considerações é possível afirmar que pode-se obter um mesmo desempenho para o sistema de agentes variando-se o método de mobilidade, as velocidades de redes e os tamanhos dos agentes. Dessa forma, deve-se definir o ambiente de hardware e software, assim como o nível de segurança e desempenho desejados, para posteriormente configurar-se os componentes para o atendimento a esses requisitos.

Os resultados demonstraram que o método de mobilidade *socketssl* possui desempenho significativamente inferior ao método *socket* para redes *Ethernet* 1000 Mbps. No entanto, optou-se por adotar a técnica *socketessl*, pois o requisito segurança é prioritário para um SDI. Esse método de mobilidade contribui para o atendimento aos princípios imunológicos e de segurança de redes estabelecidos nos Capítulos 2, 3 e 5.

7.2 MELHORIAS ADICIONAIS À FERRAMENTA

Junto das melhorias inerentes a implantação da plataforma *JADE*, como o desempenho, podemos enumerar mais alguns pontos importantes que foram acrescentados ao SDI. Entre estes, temos as seguintes características apresentadas nas próximas subseções.

7.2.1 Plataforma Iniciada a Partir do Código

Apesar de não ser uma maneira usual para iniciar a plataforma *JADE*, como esta é uma biblioteca *Java* adicionada junto ao código do programa, é possível iniciá-lá de dentro do código do SDI com os argumentos necessários. Com isto, temos um ganho na facilidade de uso da ferramenta, onde toda a configuração da plataforma já está inerente ao código, bastando o arquivo executável *Java* (extensão jar) para que a plataforma seja inicializada. Já no *Grasshopper*, era necessário instalar a plataforma no computador a ser utilizado e configurá-la anteriormente a inicialização do SDI, o que, atualmente, é feito automaticamente na execução da ferramenta. Com isto, também é facilitado a criação do contêiner remoto, no qual somente é necessário a sua execução no computador remoto, e a inserção do endereço IP do computador referente ao contêiner principal. O restante é feito de forma autônoma, desde a inicialização do *JADE*, à conexão ao contêiner principal.

7.2.2 Lista de Contêineres Atualizadas Automaticamente

Uma das modificações no SDI é a criação de um agente que fica escutando a conexão e desconexão de contêineres remotos ao contêiner principal, assim atualizando automaticamente a lista de contêineres conectados, facilitando a seleção da lista de computadores a serem parte da plataforma, onde os quais, no trabalho original, necessitavam serem adicionados manualmente a partir do endereço IP de cada computador e o tipo de *socket* de conexão.

7.2.3 Executável Multiplataforma

Com a troca de plataforma de agentes móveis, como o *JADE* é executado a partir de uma biblioteca *Java* empacotada junto ao executável, e com algumas leves alterações no código, foi possível obter uma ferramenta multiplataforma, onde independente do sistema operacional, é possível a execução da mesma, onde a única parte afetada é o comportamento de reinicialização do serviço a partir do agente anticorpo, o qual só está configurado para a reinicialização de serviços Linux. Isto possibilita uma gama maior de computadores do ambiente computacional onde é possível utilizar a ferramenta, sem necessitar, como na plataforma de agentes móveis antiga, instalar e configurar diferentemente para cada sistema operacional utilizado.

7.2.4 Segurança

Apesar de no trabalho original ser usado como medida de segurança a criptografia na conexão das agências (referente ao contêiner do *JADE*), neste trabalho atual, além da criptografia é utilizado também a autenticação dos contêineres a partir de chaves assimétricas *RSA*, onde cada contêiner tem uma *Keystore Java*, com uma chave privada, e uma *Truststore Java*, com as chaves públicas dos contêineres que serão permitidas a conexão. Assim consegue-se um melhor nível de segurança, onde não é possível que um contêiner não permitido se conecte aos contêineres originais. Com isto, garantimos além da integridade das mensagens, a autenticação e autorização dos contêineres pertencentes ao SDI.

8 CONCLUSÃO

Com este trabalho foi possível colocar em funcionamento uma importante ferramenta do DMC-NS, um Sistema de Detecção de Intrusão, onde é utilizado um Sistema Imunológico Artificial baseado no Sistema Imunológico Humano utilizando agentes móveis, desenvolvido por Machado (2005). Portanto, esta ferramenta foi modernizada, substituindo a plataforma antiga de agentes móveis, o *Grasshopper*, que se encontrava em desuso e não mais distribuída, para uma plataforma atual, o *JADE*, uma plataforma nova e com grande apelo pela comunidade. Assim, depois de pronto o desenvolvimento do trabalho, foi constatado a partir de testes, um melhor desempenho da ferramenta com a nova plataforma, além disso, foram adicionadas melhorias na ferramenta, deixando a mais segura e funcional. Um detalhe importante é que para uso, o ideal é atualizar a base de dados do *Logcheck*, para haver um melhor reconhecimento dos ataques existentes e redução dos falsos positivos e falsos negativos. A partir disto, para trabalhos futuros, deixa-se as seguintes sugestões:

- A criação de um agente representado a `Célula T-Helper`, substituindo o *Logcheck* como ferramenta necessária para o correto funcionamento do SDI, assim, melhorando sua portabilidade e facilidade de uso.
- Técnica de autoaprendizagem quanto a `antígenos self` e `non-self`.
- Técnicas de resposta para outros ambientes computacionais, como *Windows* e *Mac OS*.
- Aplicação do modelo de detecção de intrusão em um ambiente de nuvem computacional.

Estas propostas visam uma melhor eficácia do Sistema de Detecção de Intrusão, tornando mais dinâmico e melhorando a portabilidade frente a outros Sistemas Operacionais.

REFERÊNCIAS

ALLEN, J. et al. *State of the Practice of Intrusion Detection Technologies*. Pittsburgh, PA: Networked Systems Survivability Program, 2000. Seminários Ravel - CPS760: Laboratório de Redes de Alta Velocidade, UFRJ.

AXELSSON, S. *Intrusion Detection Systems: A survey and Taxonomy*. [S.l.]: New Riders, 2000.

BARBOSA, A. S.; MORAES, L. F. M. de. *Sistemas de Detecção de Intrusão*. Rio de Janeiro: [s.n.], Dezembro 2000. Seminários Ravel - CPS760: Laboratório de Redes de Alta Velocidade, UFRJ.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. *Developing Multi-Agent Systems with JADE*. Chichester, UK: Wiley, 2007. ISBN 978-0-470-05747-6.

BERNARDES, M. C. *Avaliação do Uso de Agentes Móveis em Segurança Computacional*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação (ICMC - USP), São Carlos, SP, 1999.

CAMPELLO, R. S.; WEBER, R. F. Sistemas de detecção de intrusão. In: *Anais do Simpósio Brasileiro de Redes de Computadores*. [S.l.: s.n.], 2001. Instituto de Informática UFRGS.

CANSIAN, A. M. *Desenvolvimento de um Sistema Adaptativo de Detecção de Intrusos em Redes de Computadores*. Tese (Tese de Doutorado) — Instituto de Física de São Carlos - Universidade de São Paulo, Novembro 1997.

CERT/CC. *CERT/CC Statistics 1999-2013*. Computer Emergency Response Team (Coordenation Center), Outubro 2014. Acessado em 19/03/2014. <http://www.cert.br/stats/incidentes>.

CHESS, D. M.; HARRISON, C. G.; LEBINE, D. *Itinerant Agents for Mobile Computing*. [S.l.]: IBM Research Division, 1995. IBM Research Report RC 20010.

CISCO. *Snort - A Open Source Intrusion Prevention System*. Cisco, 2014. Acessado em 16/07/2014. <https://www.snort.org/>.

COHEN, F. *Computer viruses*. [S.l.]: Computers and Security, 1987.

CROSBIE, M.; SPAFFORD, G. *Defending a Computer System using Autonomous Agents*. Tese (Doutorado) — Departament of Computer Sciences, Purdue University, 1995.

DASGUPTA, D.; FORREST, S. Artificial immune systems and their applications. In: _____. [S.l.]: Springer-Verlag Berlin and Heidelberg GmbH, 1999. cap. An Anomaly DetectionAlgorithm Inspired by the Immune System, p. 262–277.

FORREST, S.; PERELSON, A. S. Self-nonself discrimination in a computer. In: *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*. Oakland, CA: IEEE Computer Society Press, 1994. p. 202–212. <citeseer.ist.psu.edu/forrest94selfnonself.html>.

GUPTA, R.; KANSAL, G. A survey on comparative study of mobile agent platforms. *International Journal of Engineering Science and Technology*, Engg Journals Publications, v. 3, n. 3, p. 1943–1948, 2011.

HOFMEYR, S. A. An interpretative introduction to the immune system. Oxford University Press, 2000.

IKV. *Grasshopper - A Plataform for Mobile Software Agents*. IKV, 1999. Acessado em 16/11/2013. <http://www.ikv.de/products/grasshopper>.

JUCÁ, K. R. L. *Uma Abordagem de Detecção de Intrusão Baseada no Sistema Imunológico Humano*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, SC, Dezembro 2001.

KEPHART, J. O. A biologically inspired immune system for computers. In Artificial Life IV: MIT, 1994.

MACHADO, R. B. *Uma Abordagem de Detecção de Intrusão Baseada em Sistemas Imunológicos Artificiais e Agentes Móveis*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, SC, Fevereiro 2005.

NGUYEN, G. et al. Agent platform evaluation and comparison. 2002.

RAJGURU, P. V.; DESHMUKH, S. B. Analysis of mobile agent. *Journal of Global Research in Computer Science*, v. 2, n. 11, p. 6–10, 2011.

SOMAYAJI, A.; HOFMEYR, S.; FORREST, S. Principles of a computer immune system. In: DEPARTAMENT OF COMPUTER SCIENCE, NEW MEXICO UNIVERSITY, ALBUQUERQUE, NM, USA. *Meeting on New Security Paradigms, 23-26 Sept. 1997, Langdale, UK*. New York, NY, USA: ACM, 1998. p. 75–82.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 3th editon. ed. New Jersey: Prentice Hall, 2003.

TANEMBAUM, A. *Computer Networks*. 4a edição. ed. New Jersey: Elsevier, 2003.

TAVARES, D. M. *Avaliação de Técnicas de Captura para Sistemas Detectores de Intrusão*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação (ICMC - USP), São Carlos, SP, 2002.

TEIXEIRA, F. V. *JADE: Java Agent Development Framework*. Junho 2010. Trabalho da disciplina IA009 ? Introdução a Teoria de Agentes ministrada pelo professor R. R. Gudwin. Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, Campinas-SP, Brasil.

TRILLO, R.; ILARRI, S.; MENA, E. Comparison and performance evaluation of mobile agent platforms. In: IEEE. *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*. [S.l.], 2007. p. 41–41.