**Especificação e Prototipagem de um
Ambiente de Gerenciamento de Segurança
Apoiado por Agentes Móveis**

Elderclei Regis Reami

Orientação: Prof. Dr. Edson dos Santos Moreira

*Dissertação apresentada ao Instituto de Ciências Matemáticas e de
Computação – USP, como parte dos requisitos para obtenção do título
de Mestre em Ciências – Área de Ciências de Computação e
Matemática Computacional*

USP – São Carlos

Dezembro de 1998

Aos Meus Pais,
Antonio e Angela

Agradecimentos

Agradeço a todos os familiares e amigos, especialmente aos meus pais, pelo apoio dispensado durante toda a trajetória de trabalho árduo, mas gratificante, deste mestrado.

Ao professor Edson dos Santos Moreira, pelo imenso apoio dado a minha formação acadêmica, profissional e pessoal desde os primeiros semestres do meu curso de graduação até a conclusão deste trabalho e pela paciência.

Aos inumeráveis amigos de batalha, em especial: Mauricio, Laura, Mac, Taboca, Vidigal, Boni, Gustavo e Roger, entre tantos outros.

Aos amigos dos outros laboratórios de pesquisa: LABES, LCAD, LABIC, LBDPI e LASD (desculpem se esqueci de algum), em especial, aos amigos do Intermídia.

À todos aqueles que, em um momento, ou em outro, me procuraram para esclarecer suas dúvidas e, assim, me ajudaram a clarear minhas próprias.

Ao pessoal de suporte à vida durante o curso, em especial: bibliotecárias, pessoal da seção de pós-graduação (Beth, Laura e Marília), secretárias, técnicos dos laboratório e áudio-visual (Paulinho) e ao pessoal do Trembão.

À professora Maria das Graças Volpe Nunes, pela oportunidade de entrar em contato, ainda durante meu curso de graduação, com as pesquisas na área de lingüística computacional realizadas pelo convênio entre a USP e a Itautec/Philco.

À professora Maria das Graças Campos Pimentel e ao professor Dilvan de Abreu Moreira, pelo apoio e oportunidades de crescimento pessoal.

Agradeço ainda às diversas instituições públicas e privadas pelo apoio financeiro durante os cursos de graduação e de mestrado, especialmente ao CNPq.

AGRADECIMENTOS.....	III
ÍNDICE.....	IV
LISTA DE FIGURAS.....	VI
LISTA DE TABELAS.....	VI
RESUMO.....	VII
ABSTRACT	VIII
1 INTRODUÇÃO.....	9
1.1 BREVE HISTÓRICO	10
1.2 ORGANIZAÇÃO DA MONOGRAFIA	11
2 AGENTES INTELIGENTES, MÓVEIS E.....	12
2.1 DEFINIÇÃO DE AGENTE.....	13
2.2 DIFERENCIAÇÃO ENTRE AGENTES E OBJETOS	16
2.3 TIPOLOGIA DOS AGENTES	17
2.4 INFRAESTRUTURA TECNOLÓGICA	19
2.4.1 <i>Linguagens para Construção de Aplicações</i>	20
2.4.2 <i>Linguagens de Comunicação entre Agentes</i>	21
2.4.3 <i>Ontologias</i>	22
2.4.4 <i>Tecnologias de Suporte</i>	23
2.5 AGENTES MÓVEIS.....	24
2.5.1 <i>Agentes Móveis e Migração de Processos</i>	24
2.5.2 <i>Arquitetura dos Sistemas de Agentes Móveis</i>	26
2.5.3 <i>Agentes Móveis e Problemas de Segurança</i>	27
2.6 IMPLEMENTAÇÕES DE AGENTES	29
2.7 TENTATIVAS DE PADRONIZAÇÃO.....	30
2.8 SUMÁRIO	31
3 ASPECTOS DE SEGURANÇA DE COMPUTADORES.....	33
3.1 POLÍTICAS DE SEGURANÇA	34
3.2 FIREWALLS	35
3.3 SISTEMAS DE ANÁLISE DE CONFIGURAÇÃO	36
3.4 SISTEMAS DE DETECÇÃO DE INTRUSÃO	37
3.4.1 <i>Classificação dos Sistemas de Detecção de Intrusão</i>	38
3.4.2 <i>Sistemas Especialistas Baseados em Regras</i>	39
3.4.3 <i>Sistemas Baseados em Data Mining</i>	40
3.4.4 <i>Sistemas Baseados em Agentes Autônomos</i>	41
3.4.5 <i>O Sistema de Detecção de Intrusão do ICMC</i>	42
3.5 INTEGRAÇÃO DAS DIVERSAS TÉCNICAS.....	44
3.6 SUMÁRIO	44
4 GERENCIAMENTO DE SEGURANÇA APOIADO POR AGENTES MÓVEIS	45
4.1 VISÃO GERAL	46
4.1.1 <i>Uma Metáfora para o Ambiente</i>	47
4.2 ESPECIFICAÇÃO DE REQUISITOS E ANÁLISE DO DOMÍNIO	48
4.2.1 <i>Documento de Requisitos</i>	48
4.2.2 <i>Diagrama de Casos de Uso do Sistema</i>	51
4.2.3 <i>Diagramas de Classes do Sistema</i>	51

4.3 PROJETO DO SISTEMA	52
4.4 ARQUITETURA DO AMBIENTE	52
4.4.1 <i>Modelo de Distribuição de Objetos</i>	54
4.4.2 <i>Definição de uma Linguagem de Script</i>	55
4.4.3 <i>Características de Segurança</i>	55
4.4.4 <i>Ambiente de Agência</i>	56
4.4.4.1 Criação dos Agentes	56
4.4.4.2 Definição de Tarefas ou Ações	57
4.4.4.3 Transferência dos Agentes entre Ambientes de Agência	57
4.4.4.4 Transferência dos Recursos Associados	58
4.4.4.5 Comunicação entre Agentes	58
4.4.4.6 Segurança do Sistema	59
4.4.5 <i>Serviço de Persistência</i>	59
4.4.6 <i>Aplicação de Gerenciamento</i>	60
4.4.6.1 O editor de propriedades	62
4.4.7 <i>Organização da Arquitetura em Termos de Pacotes</i>	63
4.5 UM EXEMPLO DE AGENTE DE SEGURANÇA	63
4.6 INTEGRAÇÃO DO AMBIENTE SHOGUN AO NETTRACKER	64
4.7 SUMÁRIO	66
CONCLUSÃO E TRABALHOS FUTUROS	67
5.1 AVALIAÇÃO	67
5.2 TRABALHOS FUTUROS	68
REFERÊNCIAS BIBLIOGRÁFICAS	69
A - O AMBIENTE NETTRACKER	76
B - TECNOLOGIAS EMPREGADAS	79
LINGUAGENS UTILIZADAS NA PROGRAMAÇÃO DO SISTEMA	79
<i>Java</i>	79
<i>Python</i>	79
RECURSOS DA LINGUAGEM JAVA	80
<i>Introspecção ou Reflexão</i>	80
<i>RMI - Remote Method Invocation</i>	80
<i>JCA (Java Cryptography Architecture) e JCE (Java Cryptography Extension)</i>	82

Lista de Figuras

Figura 1 Contraste entre o Modelo Cliente/Servidor e o Modelo Baseado em Agente Móveis ...	13
Figura 2 Tecnologias das Aplicações Baseadas em Agentes (Nwana e Wooldridge, 1996)	20
Figura 3 Arquitetura Típica de um Sistema de Agentes ou Objetos Móveis (Vitek, Serrano & Thanos, 1997)	27
Figura 4 Arquitetura de um Sistema de Detecção de Intrusão Baseado em <i>Data Mining</i>	41
Figura 5 Arquitetura global do sistema de detecção de intrusão do ICMC.....	43
Figura 6 A Tarefa de Gerenciamento de Segurança	46
Figura 7 Diagrama de Casos de Uso do Gerente de Segurança	51
Figura 8 Diagrama de Classes do Ambiente Shogun	52
Figura 9 Arquitetura do Ambiente Shogun	54
Figura 10 Diagrama de Seqüência (Interação entre o Ambiente Shogun e o Serviço de Persistência).....	60
Figura 11 Tela de Login do Ambiente Shogun	61
Figura 12 Tela da aplicação de gerenciamento do Ambiente Shogun.....	62
Figura 13 Diagrama de Pacotes para o Ambiente Shogun	63
Figura 14 Integração do Ambiente Shogun ao NetTracker.....	65
Figura 15 Código de Integração do Ambiente Shogun ao NetTracker	65
Figura 16 Relacionamento de comunicação intramódulos	76
Figura 17 Arquitetura do sistema NetTracker	77
Figura 18 - Arquitetura de uma aplicação RMI.....	82

Lista de Tabelas

Tabela 1 Possíveis propriedades de agentes (Franklin e Graesser, 1996)	14
Tabela 2 Linguagens para Programação de Sistemas de Agentes (Nwana, 1996)	20

O crescimento do número de usuários da Internet e a abundância de informação sobre o assunto *segurança* aumentaram ainda mais os problemas relacionados à invasões de sistemas e perda de informações. Desse modo, é imprescindível que se avance rapidamente no conhecimento das técnicas de prevenção e detecção de intrusão dos sistemas. Atualmente, a grande maioria dos ambientes computacionais existentes e que implementam alguma forma de segurança faz uso de firewalls. Entretanto, o uso de firewalls, como estratégia única de defesa, pode deixar espaços para diversas formas de intrusão. Ou seja, é necessário, o uso integrado de diversas tecnologias para aumentar a capacidade de defesa de um site. Este trabalho apresenta um enfoque alternativo para o gerenciamento de segurança utilizando agentes móveis para distribuir a tarefa de monitoramento do sistema e agilizar a tomada de decisões no caso de ausência do administrador humano.

Abstract

Information about security threats are widely available on the Internet, and the number of users who are likely to use those information has been considerably increased. Therefore, there is an urge for advances in the knowledge regarding prevention techniques and intrusion detection systems. Currently, most of computing environments that implement some form of security makes use of firewalls. However, using firewalls as the only defense strategy may leave holes for several types of attacks. Therefore, the integration of several technologies is a must, so as to leverage the defensive capacity of a site. This project presents an alternative approach based on mobile agents for security management. Agents may ease the distribution of monitoring tasks, as well as speeding up the decision making process, since they are representatives of human administrators.

Capítulo

1 Introdução

O estudo da segurança em redes de computadores é uma área de crescente interesse uma vez que a rede é o meio pelo qual a maioria dos ataques ou intrusões em sistemas computacionais são realizados. Existem diversas tecnologias para se abordar este problema, dentre elas, *firewalls* e *sistemas de detecção de intrusão*. O uso destas tecnologias parte do princípio de que trocar toda a gigantesca infra-estrutura de redes e computadores inseguros é impraticável e financeiramente inviável. Diante disto, deve-se procurar meios de se aumentar a segurança de toda a infra-estrutura já existente através do uso conjunto de diversas tecnologias no sentido de proporcionar soluções altamente eficazes. Tais soluções devem ser transparentes o bastante aos usuários, não provocar grandes interferências no ambiente e fornecer mecanismos eficazes de monitoração e controle.

Em termos práticos, nem sempre é possível a permanência constante de um administrador junto aos sistemas de segurança, o que leva a necessidade por mecanismos que permitam o gerenciamento remoto. Tais mecanismos visam facilitar o trabalho do (ou da equipe do) administrador de forma que, a partir de qualquer ponto da rede, ou ainda através da Internet, ele tenha acesso às configurações e informações coletadas pelos sistemas de segurança.

Ainda, é necessário que a percepção de que algo está errado no sistema seja facilitada, pois ataques podem se dar de forma extremamente rápida. No caso de ausência do administrador, deve-se haver a possibilidade de se combater os invasores através de medidas também rápidas e isso significa rastrear os passos do intruso, produzir informações de auditoria e, certamente, interromper a ação intrusiva de alguma forma. Para isso, parece ser fundamental dar mobilidade aos sistemas detectores de intrusão e o uso de agentes móveis seria uma característica interessante para promover esta funcionalidade.

Desde 1990, diversos trabalhos nas áreas de gerenciamento e de segurança de redes foram realizados pelo Grupo de Redes e, atualmente, pelo Laboratório Intermídia. A continuidade dos projetos e a pesquisa de métodos alternativos de gerenciamento e de segurança de redes têm sido uma tônica constante. Nesse contexto, a presente proposta de trabalho representa mais uma possibilidade de expansão do conhecimento teórico e prático conseguidos pelo grupo no decorrer dos anos.

1.1 Breve Histórico

O sistema de gerenciamento de redes, que vem sendo desenvolvido pelo grupo de redes, evoluiu bastante desde o início do projeto. É possível destacar três fases bem-definidas, a saber:

1. Projeto MultiView – implementado através da integração de várias ferramentas existentes em Unix (*gawk*, *ping*, *etherfind*, etc) e estruturas de dados complexas em um ambiente gráfico bastante robusto e capaz de visualizar topologias de rede, mostrar informações de sub-redes e de equipamentos (Oda, 1994). Nessa fase, ocorreu ainda, a adição de recursos multimídia para vídeo-conferência (Liera, 1995; Moraes, 1995) e o desenvolvimento de um agente SNMP (Cicilini, 1994).
2. Projeto NetTracker – resultado da evolução do modelo de gerenciamento incluindo administração via WWW, método de extensão dos serviços do gerente e carregamento dinâmico de módulos (Mouro, 1997; Mouro, Morishita & Moreira, 1997). A implementação foi feita em linguagem Java e uma implementação de SNMPv2 deveria ter sido integrada, mas os órgãos de padronização e o mercado abandonaram aquele padrão e lançaram uma nova versão: o SNMPv3, que possui características de modularidade semelhantes ao NetTracker (Morishita, 1997).
3. A terceira fase, que está sendo realizada atualmente, consiste na integração de serviços de detecção de intrusos ao NetTracker. Um sistema de detecção de intrusos baseado em redes neurais (Cansian, 1997; Bonifácio et al., 1998) já foi desenvolvido e está sendo adaptado para que seja possível utilizá-lo através do ambiente NetTracker. A proposta do presente trabalho é adicionar características de mobilidade ao sistema de gerenciamento, bem como de realizar trocas de informações entre os sistemas, com o intuito de aumentar a chance de reconhecimento de uma invasão e acelerar a tomada de medidas por parte do gerente da rede.

1.2 Organização da Monografia

O texto desta monografia está organizado da seguinte maneira:

- O Capítulo 2 faz uma revisão ampla, mas longe de ser extensiva, sobre agentes, incluindo definições e classificação, problemas de segurança relacionados aos agentes, padrões, etc;
- O Capítulo 3 apresenta os principais aspectos de segurança relacionada a redes de computadores, fornece breve resumo das técnicas de detecção de intrusão e aborda alguns sistemas de detecção de intrusão apoiados por diversas técnicas;
- O Capítulo 4 apresenta requisitos para a implementação de um ambiente de apoio ao gerenciamento de segurança baseado em agentes e alguns detalhes de projeto e implementação do mesmo.

2 Agentes Inteligentes, Móveis e...

Durante anos os sistemas computacionais foram desenvolvidos para funcionar em plataformas centralizadas nas quais todo processamento era executado. No final da década de 80, com o advento dos equipamentos pessoais mais potentes, houve o crescimento da abordagem cliente/servidor, na qual as tarefas eram divididas entre a máquina do usuário e um, ou mais, equipamentos mais potentes que serviam de apoio para operações de banco de dados, processamento de dados, etc.

Já no início da década de 90, as corporações voltaram os olhos para o novo paradigma de análise, projeto e programação orientados a objetos que veio facilitar o desenvolvimento de sistemas e melhorar as possibilidades de reuso, aumentar a qualidade de software, etc. Além disso, o desenvolvimento de aplicações distribuídas espalhou-se, com a idéia de executar as tarefas nos locais mais adequados da rede; por exemplo, a interface com usuário é tratada no lado do cliente e o processamento de dados onde os dados estão.

Nos últimos anos, com a consolidação da tendência de adoção das tecnologias orientadas a objetos, novos paradigmas vêm sendo criados para aumentar a flexibilidade dos sistemas computacionais. Estes novos paradigmas envolvem a mobilidade dos objetos implementados no sistema e a execução assíncrona de tarefas. Mais especificamente, têm-se pesquisado muito sobre objetos móveis e sobre softwares denominados agentes móveis.

Uma das vantagens fundamentais da utilização de agentes móveis sobre a modelo cliente/servidor é a diminuição do tráfego de mensagens pela rede e, portanto, um melhor desempenho. A Figura 1 apresenta graficamente esta diferença na quantidade de comunicações ocorridas nos dois modelos.

Figura 1 Contraste entre o Modelo Cliente/Servidor e o Modelo Baseado em Agente Móveis

A tecnologia de agentes vêm sendo aplicada academicamente nos mais diversos campos e recentemente algumas aplicações comerciais vêm sendo desenvolvidas. Dentre as aplicações, desta tecnologia pode-se citar apenas como exemplos:

- Suporte para aplicações baseadas em fluxo de trabalho ou *workflow* (Nicolas, 1998);
- Gerenciamento de redes e serviços de telecomunicações (Corley et al. 1998);
- Análise de informações em aplicações de *data mining*;
- Layout de circuitos eletrônicos (Moreira & Walczowski, 1997); e
- Busca de informações em bases de dados

2.1 Definição de Agente

Em termos gerais, um agente pode ser definido como uma entidade que age em nome do usuário e com sua autorização.

Academicamente, existe muita controvérsia sobre qual seria a definição de agente. Esse fato decorre da falta de coordenação entre as diversas pesquisas paralelas que foram feitas ao longo dos anos. Por outro lado, o termo agente não é propriedade dos pesquisadores da área, sendo

usado diariamente no mundo real (*e.g.*, agente de viagem, agente econômico). De fato, segundo Nwana (1996), existe tanta chance de se atingir um consenso sobre a definição de agente, quanto dos pesquisadores de IA têm de chegar a uma definição para inteligência artificial, ou seja, nenhuma.

Por isso, seria mais interessante apresentar uma definição extremamente abrangente e uma taxonomia para os diversos tipos de agentes e funções executadas. Essa tarefa parece ter sido realizada razoavelmente por Franklin e Graesser (Franklin & Graesser, 1996), que definem, após comparação entre diversas definições de agente em vários sistemas, o conceito de agente autônomo: *“Um agente autônomo é um sistema situado em um ambiente, e parte do mesmo, que percebe aquele ambiente e age sobre o mesmo no decorrer no tempo, de acordo com sua própria agenda e de modo a afetar o que ele perceberá no futuro”*. A Tabela 1 apresenta um conjunto de propriedades citadas por Franklin e Graesser, como forma de melhorar a classificação de um agente.

Tabela 1 Possíveis propriedades de agentes (Franklin e Graesser,1996)

Propriedade	Outros Nomes	Significado
Reativo	(percepção e ação)	Responde de forma a mudanças no ambiente
Autônomo		Exerce controle sobre suas próprias ações
Orientado a metas	Com propósito pró-ativo	Não age simplesmente em função do ambiente
Temporalmente contínuo		É um processo executando continuamente
Comunicativo	Socialmente capaz	Comunica-se com outros agentes, possivelmente com humanos

Propriedade	Outros Nomes	Significado
Inteligente	Adaptável	Muda seu comportamento com base na sua experiência anterior
Móvel		Capaz de se mover de uma máquina para outra
Flexível		Ações não são definidas através de scripts
Caráter		Personalidade e estado emocional críveis

Um agente autônomo, segundo a definição genérica feita por Franklin e Graesser, sempre possui as primeiras quatro propriedades apresentadas na Tabela 1. Outros autores, como Genesereth e Ketchpel, argumentam que agentes precisam necessariamente ser comunicativos, ou seja, um programa *“é um agente de software se e somente se ele se comunica corretamente através de uma linguagem de comunicação de agentes”* (Genesereth & Ketchpel, 1994).

Outros trabalhos propõem a definição de um conceito forte de agente. Esta abordagem leva à concepção ou implementação através do uso de conceitos que são mais comumente aplicados a seres humanos. Por exemplo, é comum na comunidade de inteligência artificial encontrar agentes caracterizados por suas crenças, desejos e intenções, ou seja, através de uma noção mental de estado. Uma revisão mais ampla das idéias e de sistemas que adotam essa abordagem pode ser encontrada no trabalho de Jennings e Wooldridge (1995).

2.2 Diferenciação entre Agentes e Objetos

Para complicar a situação com relação a definição de agente, existe muita confusão entre os termos agente e objeto. Erickson (1997) chama a atenção para esta questão, citando como exemplo a diferença entre agentes de interface com o usuário e objetos de interface.

Basicamente, Erickson diferencia agente e objeto através de seus modelos conceituais de uso:

- Objetos – visíveis, passivos, possuem localização e podem conter coisas.
- Agentes – podem notar coisas, podem tomar atitudes, podem saber e aprender sobre coisas e podem se deslocar para locais diferentes.

Ainda, os usuários sabem manipular objetos de interface e não devem ter problemas ao manipular objetos novos: o usuário possui o controle total da situação. Por outro lado, os usuários sabem apenas a respeito da funcionalidade dos agentes e requisitam dados ou ações em alto nível para os mesmos: o agente possui o controle geral das operações internas e podem tomar decisões a respeito das tarefas, o que pode significar, ocasionalmente, não atender a todas as expectativas do usuário.

Embora os usuários não estejam normalmente atentos para as diferenças conceituais entre os dois modelos, Erickson justifica a separação entre objetos e agentes com a seguinte afirmação (ainda no contexto de interface gráfica): *“Objetos permanecem como eles são: coisas boas, seguras e previsíveis que apenas estão lá e tomam conta das coisas. Agentes tornam-se repositórios para funcionalidade adaptável.”*

Por outro lado, Nwana argumenta que linguagens orientadas a objetos servem melhor ao propósito de implementação de sistemas de agentes (Nwana & Wooldridge, 1996). Segundo o autor, isto ocorre por que o conceito de agentes não é tão diferente do conceito de objetos, uma vez que ambos compartilham propriedades como encapsulamento, herança e passagem de mensagens.

Portanto, apesar de muitas vezes semelhantes em termos de estrutura interna, agentes e objetos precisam ser diferenciados, como uma forma de, por exemplo, simplificar projeto de sistemas e

utilizar apenas as características realmente necessárias àquela situação. Ainda, linguagens orientadas a objetos podem facilitar a implementação de sistemas de agentes.

2.3 Tipologia dos Agentes

Agentes podem ser aplicados em diferentes contextos e em cada um desses contextos os agentes apresentam algumas características. Nwana (1996), baseando-se neste fato e na existência de diversas áreas de pesquisa utilizando o termo agente, apresenta uma classificação em várias dimensões.

Primeiramente, eles podem ser classificados segundo a sua mobilidade, ou seja, agentes podem ser estáticos ou móveis.

A segunda dimensão refere-se ao modo de funcionamento deliberativo ou reativo do agente. Um agente deliberativo possui um modelo de raciocínio simbólico interno, que permite que ele planeje e negocie tarefas com outros agentes de modo a coordenar a execução. Agentes reativos, por outro lado, apresentam um comportamento do tipo estímulo/resposta, ou seja, o agente responde ao estado atual do ambiente no qual ele está imerso.

A terceira dimensão envolve classificação de acordo com as diversas propriedades ideais e fundamentais que um agente deveria apresentar. Em seu trabalho, Nwana cita uma lista de três propriedades: autonomia, aprendizagem e cooperação.

A quarta dimensão é relacionada com o papel assumido pelo agente na aplicação, por exemplo, agentes de informação para WWW.

A quinta dimensão é a categoria dos agentes híbridos, que combinam duas ou mais das outras dimensões citadas em um único agente.

Essencialmente, os agentes se apresentam nas pesquisas e aplicações do mundo real em várias das dimensões supracitadas. No entanto, por questões de clareza, Nwana reduz a combinação dessas várias dimensões, que não seriam passíveis de representação gráfica simples, a uma lista arbitrária, mas que cobre a maioria dos tipos de agentes em investigação atualmente. Dessa forma, Nwana identifica sete tipos de agentes:

- Agentes colaboradores – enfatizam autonomia e cooperação com outros agentes para executar tarefas para seus donos. Exemplos desses agentes, incluem agentes baseados no paradigma crenças-desejos-intenções¹ sugerido por muitos autores da área de inteligência artificial distribuída;
- Agentes de interface – enfatizam autonomia e aprendizagem e a metáfora chave que dá suporte a esses agentes é do assistente pessoal que colabora com o usuário em um mesmo ambiente de trabalho. O objetivo principal é migrar da manipulação direta da interface pelo usuário para a delegação de tarefas aos agentes de interface o que facilitaria a utilização por usuários novatos, uma vez que muitas tarefas são muito complexas ou trabalhosas e exigem treinamento intensivo para sua execução correta;
- Agentes móveis – são processos capazes de “vagar” por grandes redes como a WWW, interagindo com máquinas, coletando informações e retornando após executar os deveres ajustados pelo usuário. Apesar de mobilidade não ser uma condição, nem necessária, nem suficiente para o conceito de agente, agentes móveis apresentam uma série de vantagens sobre os similares estáticos, por exemplo: redução dos custos de comunicação, independência da limitação imposta pelo uso de recursos locais, coordenação mais fácil, computação assíncrona, ambiente de desenvolvimento natural para serviços de comércio, arquitetura flexível para computação distribuída e fornecem uma nova abordagem atrativa e radicalmente diferente para o processo de projeto de aplicações;
- Agentes de informação (para Internet) – têm o papel de gerenciar, manipular e ordenar informações oriundas de várias fontes distribuídas. A necessidade para agentes deste tipo específico advém da quantidade enorme de informações disponíveis e que precisa ser pesquisada. Agentes deste tipo utilizam-se de outras tecnologias, tais como índices de busca on-line (e.g., *Altavista*, *InfoSeek*, *Lycos*,

¹ Do inglês, *BDI (belief-desires-intentions) agents*, parte importante das linhas de pesquisa em agentes que utilizam pesadamente conceitos da inteligência artificial clássica.

etc), para compilar informação sobre algum assunto requisitado pelo usuário e retornar o resultado ordenado para ele;

- Agentes reativos – representam uma categoria especial de agentes que não possuem nenhuma forma de representação simbólica interna dos seus ambientes e agem de acordo com o estado atual do ambiente em que estão imersos através de estímulos e respostas. Estes agentes normalmente são simples e se comunicam de maneiras básicas, no entanto, padrões de comportamento complexos emergem da interação entre os diversos agentes; são compostos por módulos autônomos com tarefas bem específicas e lidam com representações próximas do nível físico (e.g., dados de sensores) ao invés de representações simbólicas de alto nível. Algumas vantagens seriam robustez, flexibilidade e adaptação em contraposição à inflexibilidade, lentidão e fragilidade dos sistemas inteligentes baseados nas técnicas clássicas;
- Agentes híbridos – esses agentes baseiam-se na hipótese de que maiores benefícios resultam da utilização integrada das várias filosofias existentes. Por exemplo, um agente que implementa tanto o comportamento reativo, quanto o deliberativo, poderia apresentar respostas mais rápidas a mudanças no seu ambiente aplicando o comportamento reativo, enquanto outros problemas orientados a metas e que não exigem tal agilidade poderiam ser tratados pelo comportamento deliberativo;
- Agentes inteligentes – seriam agentes capazes de apresentar todas as características citadas possíveis de autonomia, aprendizagem e cooperação, mas que, na realidade, representam apenas aspiração dos pesquisadores atualmente.

Existem aplicações que se utilizam de dois ou mais desses tipos de agentes e Nwana refere-se a estas aplicações como sistemas de agentes heterogêneos.

2.4 Infraestrutura Tecnológica

Embora não exista uma forma de predizer qual será a forma futura das aplicações e da tecnologia de agentes, existe uma série de questões que aparecem frequentemente em todos sistemas de

agentes atuais e, por outro lado, vários problemas precisam ser resolvidos por todos projetistas de sistemas de agentes. Nwana e Wooldridge (1996) apresentam uma organização abstrata das tecnologias que habilitam a construção de aplicações baseadas em agentes.

Figura 2 Tecnologias das Aplicações Baseadas em Agentes (Nwana e Wooldridge, 1996)

Em termos de linguagem, é possível distinguir dois tipos específicos: linguagens para construção das aplicações e linguagens de comunicação entre agentes. As ontologias suplementam a função de troca de conhecimento das linguagens de comunicação entre agentes. As tecnologias de suporte fornecem meios de avançar o paradigma baseado em agentes.

Os tópicos a seguir discutem um pouco mais amplamente as tecnologias referidas que formam a infraestrutura capaz de suportar aplicações baseadas em agentes.

2.4.1 Linguagens para Construção de Aplicações

Linguagens para construção de aplicações são as mais diversas possíveis, no entanto, como já observado anteriormente, linguagens orientadas a objetos servem melhor ao propósito de implementação de sistemas de agentes.

A Tabela 2 adaptada de Nwana (1996) apresenta uma seleção não extensiva de algumas linguagens e as relaciona com os tipos de agentes a que mais se adaptam.

Tabela 2 Linguagens para Programação de Sistemas de Agentes (Nwana, 1996)

Tipo de Agente	Classe de Linguagem	Exemplos
Agentes Colaboradores	Linguagem de Atores	Actors
	Linguagens de Programação Orientadas a Agentes	Agent-0 Placa
Agentes de Interface Agentes de Informação Agentes Móveis	Linguagens de <i>Script</i>	TCL/Tk Safe-TCL, Safe-Tk Java Telescript Python Obliq April Scheme-48
Agentes Reativos	Linguagens Reativas	RTA/ABLE

Em seu artigo, Nwana fornece uma breve descrição e referências bibliográficas para cada uma das linguagens apresentadas na Tabela 2.

2.4.2 Linguagens de Comunicação entre Agentes

Uma característica que, geralmente, distingue agentes de outros tipos de programas e objetos é a capacidade de troca de informação e conhecimento entre os agentes que fazem parte de uma comunidade. Dessa forma, a colaboração entre agentes torna-se uma vantagem relevante, pois, através da troca de conhecimento, uma comunidade de agentes pode realizar melhor as tarefas relativas as suas metas intrínsecas.

No entanto, para que haja troca de conhecimento entre os agentes é preciso que haja um esforço extra na representação do mesmo, bem como é necessário definir uma linguagem para comunicação deste conhecimento entre os agentes.

Diversas entidades estão trabalhando com intuito de resolver o problema de representação e compartilhamento de conhecimento. Algumas soluções são:

- KQML (*Knowledge Query and Manipulation Language*), ou Linguagem de Manipulação e Consulta de Conhecimento – concebida como um formato de mensagem e um protocolo para tratamento de mensagens para suportar a troca em tempo de execução de conhecimento entre os agentes (Finin, Labrou & Mayfield, 1997).
- ACL (*Agent Communication Language*), ou Linguagem de Comunicação entre Agentes – proposta pela FIPA e, fortemente, baseada na idéia dessa instituição do que seria uma teoria de agentes adequada (FIPA, 1997).

2.4.3 Ontologias

Para que haja comunicação efetiva entre os agentes, além de uma linguagem de comunicação, é necessário que os agentes utilizem o mesmo vocabulário para expressar os objetos e relações dentro do domínio da aplicação. Isto significa que os agentes devem compartilhar o mesmo tipo de conhecimento no domínio da aplicação.

Para tornar essa tarefa possível devem ser criadas ontologias. Segundo Gruber (1993a), uma ontologia é uma especificação explícita de uma concepção. Uma concepção reflete uma visão simplificada do mundo que se quer representar para algum propósito.

O termo ontologia tem origem na filosofia, na qual ele é uma indicação sistemática de existência. Por sua vez, para os pesquisadores de inteligência artificial, o que existe é o que pode ser representado.

Para representar ontologias de forma independente de dados específicos ou linguagens de programação, foram definidas linguagens declarativas. Provavelmente, os trabalhos mais avançados nesta área foram feitos pelo *Knowledge Sharing Effort* (KSE) do ARPA. Os trabalhos do KSE incluem:

- KIF (*Knowledge Interchange Format*), ou Formato de Intercâmbio de Conhecimento – o KIF fornece uma linguagem intermediária para que bases de conhecimento possam interoperar e consiste, basicamente, em notação prefixada para cálculo de predicados de primeira ordem com algumas extensões para aumentar sua expressividade. A aparência de uma sentença expressa em KIF é semelhante a um trecho de código em LISP. Maiores detalhes podem ser encontrados em (Genesereth & Fikes, 1992).
- *Ontolingua* – representa um esforço do ARPA com o intuito de criar ontologias reutilizáveis. É uma ferramenta de tradução independente do domínio, que permite a conversão de ontologias prontas (*off-the-shelf*) em diversas representações simbólicas. Maiores detalhes podem ser encontrados em Gruber (1993b) e na página da Ontolingua na WWW².

Outros esforços vêm sendo realizados por outras instituições com a intenção de criar ontologias globais de uso geral. Dois exemplo notáveis citados por Nwana e Wooldridge (1996) são os projetos WordNet e Cyc, mas uma discussão mais aprofundada dos mesmos está fora do escopo deste texto.

2.4.4 Tecnologias de Suporte

As tecnologias de suporte citadas por Nwana (1996) são relacionadas principalmente ao uso de diferentes meios de distribuição pelos sistemas de agentes, ou seja, uso do modelo cliente/servidor e de tecnologias que transformam os agentes em agente móveis. As tecnologias e implicações do uso de agentes móveis serão discutidas nas próximas seções.

²A página principal da Ontolingua está localizável em <http://ksl-web.stanford.edu/knowledge-sharing/ontolingua/>

2.5 Agentes Móveis

O conceito de código móvel, que suporta os agentes móveis, não é novo. Exemplos recentes são a submissão remota de trabalhos em lote e a linguagem *PostScript* para controle de impressoras e dispositivos gráficos.

As pesquisas sobre os mecanismos sofisticados que suportam a mobilidade foram avançadas, em grande parte, pelo esforços da comunidade de sistemas operacionais distribuídos.

Diversos trabalhos foram realizados com o intuito de avaliar a factibilidade e vantagens do uso de agentes móveis. Um dos trabalhos mais citados em toda literatura é o relatório de pesquisa feito por Harrison, Chess e Kershenbaum (1997), que foi posteriormente publicado como artigo também. Esse trabalho apresenta as possíveis vantagens e desvantagens do uso de agentes móveis, comparando-os com técnicas tradicionais de implementação de sistemas distribuídos como RPC.

Fuggeta, Picco e Vigna (1998) citam diversas aplicações que poderiam se beneficiar do uso de código móvel, em particular, de agentes móveis. Entre elas estão destacadas: busca de informação distribuída, documentos ativos, serviços de telecomunicação avançados, configuração e controle de dispositivos remotos, gerenciamento de fluxo de trabalho (*workflow*) e cooperação, redes ativas e comércio eletrônico. As próximas seções discutem alguns dos aspectos relacionados aos agentes móveis.

2.5.1 Agentes Móveis e Migração de Processos

Milojicic, Guday e Wheeler afirmam que a necessidade de se ter agentes, que migram entre nós com arquiteturas e sistemas operacionais diferentes, favorece a mudança do suporte à mobilidade do nível do sistema operacional para o nível das linguagens de programação, mas a rica experiência com os mecanismos ao nível de sistema operacional deveria ser utilizada para guiar estas implementações de mais alto nível (Milojicic, Guday & Wheeler, 1997).

Apesar das razões específicas para ocorrer a migração em cada uma das abordagens serem diferentes (e.g., balanceamento de carga *versus* necessidade de obter informações especiais), o estado da técnica e da arte necessários para enfrentar os problemas resultantes daquela migração são semelhantes.

Dessa forma, é possível enumerar as seguintes características dos sistemas de migração de processos baseados no sistema operacional e aplicáveis aos ambientes de agentes móveis:

- nomeação, endereçamento e localização de processos;
- controle da migração de processos;
- exportação do estado do processo;
- transferência de dados do processo;
- comunicação transparente;
- sistemas de arquivo;
- segurança.

Por outro lado, linguagens de programação como Java, Tcl e Python, têm provado a sua adequação para implementação de ambientes de agentes móveis, pois fornecem, entre outras características: mobilidade ao código, independência de sistema operacional e simplicidade semântica. Entretanto, agentes móveis também necessitam das características de nomeação, endereçamento e localização, controle de mobilidade, exportação do estado de execução do agente e segurança; que muitas vezes são deficientes, ou não estão presentes, nas linguagens citadas.

Outros aspectos que devem ser considerados quando se compara o suporte à migração de processos fornecida por sistemas distribuídos fortemente acoplados e o suporte à mobilidade dos sistema de agentes móveis são apresentados em (Fuggeta, Picco & Vigna, 1998):

- Mobilidade explorada na escala da Internet: os sistemas operacionais distribuídos suportam mobilidade em redes de computadores de pequena escala, supondo que estão disponíveis grande banda de passagem, tempo de latência pequeno e previsível, confiabilidade e, freqüentemente, homogeneidade. Por outro lado, os sistemas de agentes móveis são concebidos para operar em configurações radicalmente diferentes destas;

- Programas estão conscientes da sua localização atual: enquanto nos sistemas distribuídos convencionais os processos ignoram qual é o seu ambiente atual de execução, aplicações baseadas em código móvel sabem exatamente onde estão executando e podem tomar decisões com base neste conhecimento;
- Mobilidade está sob controle do programador: o programador tem meios de transferir ou buscar fragmentos de código de e para máquinas remotas. O sistema de suporte a execução (ambiente computacional) fornece os meios para que aquelas operações sejam realizadas, mas não tem nenhum controle sobre as políticas de migração;
- Mobilidade não é apenas um recurso para balanceamento de carga: como já enfatizado anteriormente, a migração de processos tem como meta principal o balanceamento de carga, enquanto os sistemas de agentes móveis visam atender a um conjunto muito maior de necessidades e requisitos, como configuração de serviços, extensão dinâmica de aplicações, autonomia, tolerância à falhas e suporte a operação *off-line*.

Dessa forma, apesar do conhecimento agregado pelas pesquisas sobre migração de processos serem fundamentais para a implementação de sistemas de objetos ou agentes móveis, estes últimos possuem um escopo muito maior de aplicação e, portanto, superam a migração de processos convencional em termos de funcionalidade.

2.5.2 Arquitetura dos Sistemas de Agentes Móveis

Em termos de organização, a arquitetura de um sistema de agentes (ou objetos) móveis é composta por quatro componentes: o *host* – um computador e sistema operacional, o ambiente computacional (AC) – o sistema de execução (*run-time system*), sistemas de agentes (ou objetos) móveis – o conjunto de computações sendo executadas em um AC, e a rede ou subsistema de comunicação que interliga os vários ACs rodando em *hosts* diferentes (Vitek, Serrano & Thanos, 1997). A Figura 3 mostra a arquitetura de um sistema de agentes móveis.

Figura 3 Arquitetura Típica de um Sistema de Agentes ou Objetos Móveis (Vitek, Serrano & Thanos, 1997)

Os componentes da arquitetura interagem uns com outros para implementar os comportamentos dos agentes e dependendo das permissões e restrições associadas a essas interações podem ocorrer problemas de segurança.

2.5.3 Agentes Móveis e Problemas de Segurança

Outro aspecto importante relacionado aos agentes e ao seu uso é a questão da segurança. Parece claro que um vírus de computador pode ser caracterizado também como um agente, uma vez que ele possui pelo menos as quatro primeiras propriedades da Tabela 1 e principalmente mobilidade. Parece correto dizer que se um agente é móvel, ele pode ser responsável por causar graves problemas de segurança.

Dessa forma, a questão da segurança relacionada aos agentes e, em especial, aos agentes móveis é de suma importância.

Sander e Tschudin (1998a) fornece um exemplo interessante do tipo de problema de segurança que pode ocorrer durante uma busca por preços de passagens feita por um agente de compras. Suponha que o objetivo programado pelo usuário seja visitar os servidores de várias empresas aéreas, encontrar um voo disponível e, uma vez determinada a melhor oferta, agendar o voo. Alguns ataques possíveis nesta situação são:

- ao chegar em um servidor A (*malicious host*), o estado e código do agente são alterados de forma que ele esqueça os outros servidores já visitados e opte pela oferta do servidor A;
- alterar o estado interno do agente e roubar toda moeda eletrônica que ele está levando;
- agente resolve agendar o voo e, portanto, precisa assinar digitalmente através de sua chave-privada: a chave-privada é roubada.

Observando a Figura 3, é possível verificar as diversas fronteiras em que ocorrem os problemas de segurança relacionadas com os sistemas de agentes. Vitek, Serrano e Thanos (1997) classificam aqueles problemas e propõe algumas soluções possíveis:

- comunicação entre *hosts* e ACs precisa ser segura para prevenir a revelação de informações e a corrupção de dados e códigos;
- computações recebidas devem ser autenticadas e ter direitos de acesso garantidos, para que se saiba quem está executando a computação, quem deve ser cobrado, etc;
- acesso aos recursos locais do *host* deve ser controlado;
- o AC deve ser protegido de computações possivelmente maliciosas (e vice-versa), para se evitar que as computações passem por cima das regras de segurança do sistema;
- os sistemas de objetos móveis devem ser protegidos uns dos outros, para prevenir a interrupção de uma computação por outra e para evitar que uma consiga informações privilegiadas da outra.

Os problemas mais preocupantes, no entanto, são: proteger um *host* de um agente mal intencionado e proteger um agente de um *host* mal intencionado. O primeiro problema pode ser tratado através de técnicas convencionais como listas de controle de acesso e autenticação. Por sua vez, é amplamente aceito que não existe uma solução que previna a ocorrência do segundo problema, a menos que haja *hardware* confiável e a prova de adulteração (Chess et al., 1995).

No entanto, vários autores estão trabalhando com a hipótese de que seja possível proteger agentes de *hosts* mal intencionados sem a necessidade daquele tipo de hardware.

Diversas pesquisas vêm sendo feitas com o intuito de criar ambientes de agentes que possam garantir um mínimo de segurança. No entanto, os maiores problemas localizam-se na tarefa de proteger os agentes da ação de máquinas mal-intencionadas (Vitek & Tschudin, 1997; Sander & Tschudin, 1998b).

2.6 Implementações de Agentes

Diversos sistemas de agentes móveis e *frameworks* de suporte ao desenvolvimento de aplicações que utilizam esta tecnologia vêm sendo propostos e implementados academicamente. Apenas como forma de citação, alguns exemplos são:

- ARA – visa criar uma plataforma para execução segura e portátil de agentes móveis em redes heterôgeneas;
- D'Agents (antigo Agents TCL) – um sistema de agentes móveis para busca, distribuição e apresentação de informação distribuída em redes arbitrárias;
- JAT (*Java Agents Template*) – um esqueleto totalmente funcional escrito em Java para construção de agentes, que se comunicam de forma fim-a-fim com comunidades de outros agentes distribuídos pela Internet. Apesar desse ser um *framework* no qual os agentes são estáticos, mobilidade pode ser adicionada facilmente através do uso de serialização de objetos e RMI (*Remote Method Invocation*), além disso, esse *framework* fornece suporte para KQML e troca de recursos entre os agentes, entre outras características interessantes;
- Sumatra – uma extensão da linguagem e da máquina virtual Java para suportar objetos e processos móveis, que pode ser utilizada para programar sistemas de agentes móveis;
- TACOMA (Tromsø and Cornell University System) – projeto conjunto entre essas universidades sobre os aspectos dos sistemas operacionais para suportar agentes em rede.

Outras implementações comercialmente disponíveis de sistemas de agentes móveis, com maior ou menor grau de compromisso com os conceitos citados nas seções anteriores, seriam Aglets criado pela IBM, Concordia da Mitsubishi, Odissey da General Magic e Voyager criado pela ObjectSpace, entre outros.

2.7 Tentativas de Padronização

Apesar de não haver ainda um consenso geral a respeito do que são e de quais são as características fundamentais dos agentes, diversas entidades estão tentando formular definições, criar modelos de referência e extensões com o intuito de padronizar esta tecnologia. As entidades mais reconhecidas atualmente são:

- *Agents Society*³ – é uma entidade internacional formada por profissionais e indústrias, que foi estabelecida para promover o desenvolvimento e crescimento do mercado e da tecnologia de agentes inteligentes móveis. Algumas atividades da *Agents Society* são: facilitar a colaboração e a transferência de tecnologia através da coleção e troca de informações sobre os desenvolvimento em agentes; a criação de padrões abertos para agentes e interoperabilidade;
- FIPA⁴ (*Foundation for Intelligent Physical Agents*) – é uma associação internacional sem fins lucrativos de companhias e organizações. Seu objetivo é promover a especificação e o desenvolvimento de tecnologias de agentes, que sejam utilizáveis por uma grande diversidade de aplicações resultando em um alto grau de interoperabilidade entre as aplicações. Alguns aspectos que estão sendo tratados com fins de padronização são: comunicação, representação da informação, sociedades de agentes, ambiente de execução para agentes móveis e gerenciamento de agentes;

³ A página da *Agents Society* está localizável em <http://www.agents.org>

⁴ A página da FIPA está localizável em <http://drogo.cselt.stet.it/fipa/>

- **OMG⁵ (Object Management Group)** – é a entidade responsável pela especificação CORBA (*Common Object Request Broker Architecture*) para aplicações que utilizam objetos distribuídos em ambientes heterôgeneos e das tecnologias relacionadas a essa arquitetura. Em termos de mobilidade, o OMG está estudando a criação de dois padrões: MAF (*Mobile Agents Facility*) e MOF (*Mobile Objects Facility*);
- **W3C⁶ (World-wide Web Consortium)** – é a entidade responsável pela determinação dos padrões utilizados na WWW, incluindo linguagens de marcação, estilos e metadados, protocolos, etc.
- **IETF⁷ (Internet Engineering Task Force)** – são os responsáveis pelo estudo e criação dos padrões de protocolo básicos que movem a Internet como um todo, por exemplo, TCP, IPv6, RSVP, etc.

Como em outras áreas (*e.g.*, telecomunicações e sistemas distribuídos), a criação de padrões é fundamental para permitir que os agentes possam cooperar em tarefas e para que o usuário possa utilizar sistemas de fornecedores diferentes de forma transparente. A respeito dessa questão, Nwana e Wooldridge (Nwana & Wooldridge, 1996) se pronunciam da seguinte maneira: “*Os padrões serão essenciais, se for esperado que a tecnologia de agentes realize todo seu potencial...*”.

2.8 Sumário

O uso do termo agente no mundo da computação e da inteligência artificial tem sido diverso e, muitas vezes, o termo é usado erroneamente. No entanto, a medida que a tecnologia está avançando algumas características fundamentais dos agentes vêm se destacando. Em particular,

⁵ A página do OMG está localizável em <http://www.omg.org>

⁶ A página do W3C está localizável em <http://www.w3c.org>

⁷ A página do IETF está localizável em <http://www.ietf.org>

reatividade, autonomia, orientação à metas e continuidade temporal têm sido consideradas características importantes.

Apesar de muitos considerarem que ainda não surgiram aplicações que necessitem diretamente do uso específico de agentes, muitas aplicações têm sido desenvolvidas utilizando essa abordagem e alguns dos resultados têm sido: modelagem facilitada, melhor projeto, escalabilidade, etc.

A criação de padrões para o uso de agentes poderá fornecer um impulso à adoção dos mesmos nos mais diversos campos de aplicação, porém, muito ainda deve ser realizado até que seja possível a concordância geral sobre esta tecnologia.

3 Aspectos de Segurança de Computadores

Em termos de segurança, dois aspectos devem ser levados em consideração: a segurança de comunicações e a segurança de sites. O aspecto segurança de comunicações não vai ser detalhado neste texto, pois trata-se de um assunto extenso e sem muito interesse para o tópico detecção de intrusão.

A segurança do *site* diz respeito à segurança dos recursos computacionais presentes em uma rede privada. Tais recursos são compostos por *hosts*, roteadores, impressoras, informações armazenadas, servidores de banco de dados e softwares em geral.

Quatro princípios básicos definem a segurança de um site:

- **Confidencialidade:** Apenas quem tem os direitos de acesso a um determinado recurso ou informação poderá efetivamente utilizá-los.
- **Integridade:** Garante que os dados armazenados não serão alterados, tanto como consequência de atos provenientes de intrusão, quanto a eventos como quedas de energia e falhas nos sistemas.
- **Disponibilidade:** Garante que os recursos computacionais e os dados presentes neles estarão disponíveis sempre que necessários. Atualmente, um número cada vez maior de ataques exploram furos que causam falhas na disponibilidade dos sistemas. Tais ataques são geralmente chamados de *denial of service*.
- **Autenticação:** Diz respeito à identidade de um usuário, ou seja, garantir se o usuário realmente é quem diz ser.

Os administradores de sistemas devem considerar esses quatro princípios básicos e definir uma política de segurança adequada.

3.1 Políticas de Segurança

A política de segurança irá definir o que pode ou não ser feito, por quem, sob quais circunstâncias, define os procedimentos (Plano de Emergência) a serem tomados frente a uma invasão, quais são os recursos que devem ser prioritariamente protegidos, que nível de risco é aceitável, além de resolver questões éticas e polêmicas como, por exemplo, se o administrador ou um chefe tem o direito de ler o e-mail dos funcionários. Uma vez que se tenha tudo definido e que cada usuário e administrador esteja ciente desta política, tem-se um quadro onde é fácil identificar que ações são consideradas como uma quebra de segurança e que medidas podem ser tomadas.

Uma quebra de segurança pode ser definida como uma ação, ou conjunto de ações, que vão contra a política de segurança vigente. Esta política de segurança é criada com as necessidades particulares de cada local. Desta forma, o que é considerado uma quebra de segurança pode variar conforme o local.

Uma quebra de segurança pode ocorrer onde existir uma falha. Falhas podem ser atribuídas a três causas principais:

- **Softwares:** Os softwares que rodam nos computadores podem apresentar falhas que podem ser exploradas por atacantes, ou ainda, falhas que podem prejudicar a rede, como por exemplo, um servidor de banco de dados mal projetado que perca informações. Por outro lado, um software, devido à sua procedência duvidosa, pode conter *backdoors* ou ser um *Trojan Horse* (Cavalo de Tróia). O outro problema com softwares, como dito anteriormente, é a demora dos fabricantes em lançar *patches* de segurança de problemas que tenham sido descobertos.
- **Administradores:** A menos que se tenha uma pessoa com a função específica de administrador de segurança, este será outro grande problema. Geralmente, administradores não dão a devida importância à segurança da rede, quer seja por falta de informação ou por desinteresse. As falhas mais comuns oriundas de administradores são: não instalam sistemas de proteção e auditoria; não aplicam *patches* de problemas conhecidos; negligenciam procedimentos básicos de segurança; não orientam os usuários e não se mantêm atualizados.

- **Usuários:** Este é outro grande perigo para a segurança de uma rede: usuários mal informados ou mal intencionados podem causar grandes prejuízos. Um ataque se torna muito mais fácil e com muito mais chances de sucesso se proferido por um usuário do próprio sistema com intenções de roubo de informações ou até mesmo vingança contra um colega ou um superior. Usuários comuns podem expor o sistema com senhas fracas, podem fornecer suas senhas para terceiros, utilizar programas de origem duvidosa e muitas outras ações que podem ir contra a política de segurança ou que não estejam previstas, mas que podem colocar a rede sob perigo.

3.2 Firewalls

O propósito de um *firewall* é prevenir o acesso de usuários não autorizados aos recursos computacionais em uma rede privada e a exportação não notificada e não autorizada de informações confidenciais.

Na configuração de um *firewall*, as principais decisões relacionadas à segurança são freqüentemente ditadas pela política da organização ou corporação. Especificamente, as decisões devem ser tomadas pesando-se até que nível a segurança deve ser mais importante que a flexibilidade e facilidade de uso dos recursos computacionais que o *firewall* se destina a proteger.

Há duas abordagens básicas na configuração de um *firewall*:

- *O que não é expressamente proibido é permitido*
- *O que não é expressamente permitido é proibido*

No primeiro caso, o administrador do sistema tem de prever que tipos de ações os usuários, ou pessoas externas, podem fazer que infringem a política de segurança; e preparar defesas contra elas. No segundo caso, o *firewall* deve ser projetado para bloquear tudo, os serviços devem ser permitidos caso a caso após um cuidadoso estudo de necessidade e risco. Isto causa impacto imediato nos usuários, que podem ver o *firewall* como um estorvo. Esta opção é também a mais segura, já que o administrador não precisa conhecer profundamente que portas TCP são seguras ou que furos de segurança podem existir no *kernel* do sistema ou nas

aplicações. Uma vez que muitos fabricantes demoram em publicar furos de segurança, esta é claramente uma abordagem mais conservadora, com base no fato de que o que você não conhece pode ser perigoso para você. Esta segunda abordagem pode ser bem empregada em empresas com normas bem definidas e rígidas, porém pode-se tornar altamente imprópria em instituições de ensino e pesquisa, uma vez que ela restringe em demasia o uso dos recursos computacionais. A não ser que se tenha uma equipe muito bem preparada e que reconheça a fundo todas as diferentes necessidades dos usuários do sistema e reflitam isto na configuração, o *firewall* pode ser tornar um inconveniente e mesmo prejudicar o andamento de pesquisas.

Diversos esquemas e configurações para *firewalls* são apresentados de forma aprofundada na literatura pesquisada. Cada uma das configurações possíveis possuem pontos fortes e fragilidades específicas e informações mais detalhadas podem ser encontradas em (Chapman, 1992; Wack & Camahan, 1994; Bellovin & Cheswick, 1994).

3.3 Sistemas de Análise de Configuração

Sistemas de análise de configuração foram desenvolvidos para permitir a verificação de vulnerabilidades comuns, como, por exemplo, sistemas de arquivo exportados indevidamente e permissões incorretas em arquivos e diretórios sensíveis a problemas de segurança.

A necessidade por tais sistemas advém do fato de que a configuração correta dos parâmetros de controle de acesso de um sistema dependem de decisões complexas e que influem umas nas outras. Por isso, é possível que um administrador menos experiente configure um sistema de forma incorreta ou incompleta, daí a necessidade de verificação da configuração atual.

Sistemas de análise de configuração realizam uma análise sistemática das configurações de um computador, de modo a descobrir possíveis furos de segurança que podem tornar um sistema susceptível a invasões, uso indevido ou abuso.

Diversas ferramentas de análise de configuração foram desenvolvidas e são usadas periodicamente para julgar a vulnerabilidade de sistemas a ataques:

- COPS (*Computer Oracle Password and Security System*)⁸ – ferramenta que cria um relatório de vulnerabilidades comuns como senhas fracas, diretórios de usuários com permissões de escrita globais, exportação indevida de sistemas de arquivos, etc (Farmer & Spafford, 1990);
- SATAN (*Security Administrator Tool for Analyzing Networks*) – desenvolvido para identificar serviços de rede vulneráveis em um determinado domínio da rede (Venema, 1995);
- NetKuang – com função similar ao SATAN (Zerkle & Levitt, 1996);
- Miro – um sistema que usa notação visual para descrever configurações e políticas de segurança de um sistema de arquivos (Heydon, 1992);
- TripWire – um sistema que armazena informações de todos arquivos do sistema em uma base de dados através de assinaturas digitais e que permite que essas informações sejam posteriormente verificadas quanto a alteração ou corrupção (Kim & Spafford, 1994).

Além dos sistemas citados, vários outros vêm sendo desenvolvidos e alguns tornando-se comercialmente disponíveis, como o ISS (Internet Security Scanner) (ISS, 1995) e o próprio Tripwire.

3.4 Sistemas de Detecção de Intrusão

Apesar do uso dos diversos esquemas de segurança existentes, ainda existe a possibilidade de ocorrência de falhas nestes esquemas e, portanto, é desejável a existência de sistemas capazes de realizar a detecção de tais falhas e informar o administrador da rede. Tais sistemas são conhecidos como sistemas de detecção de intrusão (SDI).

⁸ Tanto COPS como NetKuang são baseados sobre um sistema de verificação de segurança baseado em regras conhecido como SU-KUANG (Baldwin 1994), que adota o raciocínio de um atacante para procurar por furos no sistema.

Tentativas de ataque acontecem de acordo com algumas técnicas de acesso e frequentemente o invasor está fisicamente fora do sistema sob ataque. Os primeiros modelos de sistemas de detecção de intrusão projetados para computadores isolados usavam algoritmos básicos que incluem análise de funções multinomiais e aproximação de matrizes covariantes para detectar desvio do comportamento normal, bem como sistemas especialistas para detectar violação de políticas de segurança. Os modelos mais recentes monitoram um grande número de redes de computadores e transferem a informação monitorada para ser processada em um equipamento central que emprega técnicas de sistemas distribuídos.

A maioria dos SDIs tem um processo auditor (*daemon*) em cada máquina, responsável por capturar ações de violação de segurança dentro da máquina. Sistemas baseados em redes, ao invés de utilizar pistas de auditoria, analisam o tráfego de pacotes dentro da rede para detectar comportamento intrusivo.

As funcionalidades de um sistema de detecção tornam-se de vital importância na medida em que fornecem meios de inferir sobre o conteúdo das conexões permitidas e detectar as que apresentem um comportamento suspeito ou não condizente com a política de segurança implantada.

3.4.1 Classificação dos Sistemas de Detecção de Intrusão

Os SDI são classificados sob diferentes ópticas na literatura da área. As principais classificações utilizadas são em termos da forma como o sistema aborda o problema de detectar intrusão e em termos do tratamento dos dados.

As três principais abordagens para detecção de intrusão são: detecção de anomalias, detecção de uso indevido e detecção híbrida. Sistemas de detecção de anomalias definem um modelo de atividade normal através de perfis de atividade dos usuários e qualquer desvio significativo da norma estabelecida é considerada anômala. Por sua vez, sistemas de detecção de uso indevido definem especificamente que ações do usuário podem constituir um uso indevido do sistema e usam regras definidas *a priori* para codificar e detectar padrões de intrusão conhecidos. Cada uma das abordagens é mais adequada para certos tipos de ataque e, por isso, muitos sistemas utilizam uma abordagem híbrida para tornar a detecção mais eficiente.

Em termos do tratamento dos dados, existem sistemas baseados em rede e sistemas baseados em *host*. Sistemas baseados em rede examinam os dados que trafegam pela rede através da monitoração *on-line* dos pacotes. Por outro lado, sistemas baseados em *host* analisam dados de auditoria recolhidos normalmente pelos sistemas operacionais e buscam de diversas formas pelos padrões de ataque.

As próximas seções apresentam apenas alguns dos muitos SDIs existentes e que utilizam diversas abordagens para resolver o problema de detecção.

3.4.2 Sistemas Especialistas Baseados em Regras

Sistemas especialistas baseados em regras são sistemas inteligentes que capturam conhecimento de especialistas humanos em domínios limitados do conhecimento, geralmente, na forma de regras do tipo IF-THEN. Alguns trabalhos vêm sendo realizados para utilizar esta técnica de inteligência artificial em sistemas de detecção de intrusão.

O SRI International desenvolveu um dos primeiros sistemas baseados nesta técnica o IDES (*Intrusion Detection Expert System*) apresentado em (Lunt & Jagannathan, 1988). A evolução deste sistema foi o NIDES (*Next Generation Intrusion Detection Expert System*) (Javitz et al., 1993).

Atualmente, o SRI está desenvolvendo o EMERALD (*Event Monitoring Enabling Responses to Anomalous Live Disturbances*) como sucessor do NIDES. O EMERALD representa uma evolução das pesquisas anteriores com o intuito de acomodar a monitoração de grandes redes e sistemas distribuídos. Este sistema apresenta uma arquitetura altamente modular e permite a integração de novas características facilmente, inclusive a integração de sistemas fornecidos por terceiros (Porras & Neumann, 1996; 1997).

3.4.3 Sistemas Baseados em *Data Mining*⁹

Lee e Stolfo (Lee & Stolfo, 1997) apresentam uma abordagem baseada em técnicas de *data mining* para descobrir padrões úteis e consistentes das características do sistema que descrevem os comportamentos de usuários e programas. A partir daí, o conjunto de características relevantes do sistema é usado para computar classificadores (aprendidos por indução) que podem reconhecer anomalias e intrusões.

No trabalho referenciado, Lee e Stolfo afirmam que o maior desafio para a utilização de técnicas de *data mining* na detecção de intrusão é a imensa quantidade de dados de auditoria necessários para computar os conjuntos de perfis de uso do sistema. Como o processo de aprendizado (*mining*) é caro em termos de tempo e de armazenamento; e como o processo de detecção em tempo real precisa ser leve e rápido para ser praticamente utilizável, seria impossível de se ter um sistema de detecção de intrusão monolítico. Assim, eles propõem uma arquitetura na qual existem dois tipos de agentes inteligentes: agentes de aprendizado e agentes de detecção. A Figura 4 apresenta a arquitetura sugerida por Lee e Stolfo.

⁹ Apesar de alguns autores utilizarem a tradução “Mineração de Dados” para o termo *Data Mining*, neste texto será mantida a versão em inglês, pois ela é bem mais utilizada pelos autores da área no Brasil do que a versão traduzida.

Figura 4 Arquitetura de um Sistema de Detecção de Intrusão Baseado em *Data Mining*

3.4.4 Sistemas Baseados em Agentes Autônomos

A utilização de agentes autônomos têm sido proposta por alguns autores como forma de construir sistemas de detecção de intrusão não-monolíticos. A capacidade dos agentes autônomos de manter informação específica do seu domínio de aplicação, nesse caso, aplicação de segurança, dá a estes agentes grande flexibilidade.

Crosbie e Spafford especificam um sistema deste tipo no qual as capacidades dos agentes são modificadas através do uso algoritmo genéticos. Os autores reconhecem, entre outras, as seguintes vantagens de sistemas baseados em agentes autônomos sobre sistemas monolíticos (Crosbie & Spafford 1995a; 1995b):

- fácil configuração: uma vez que é possível ter uma série de pequenos agentes especializados em tarefas específicas de detecção, o sistema de detecção pode ser

configurado da forma mais adequada para cada caso; a adição e remoção de agentes do sistema é facilitada;

- eficiência: agentes podem ser treinados previamente e otimizados para que realizem suas tarefas da maneira a gerar a menor sobrecarga do sistema possível;
- capacidade de extensão: um sistema de agentes pode ser facilmente modificado para operar em rede e permitir migração para rastrear comportamentos anômalos através da rede, ou mover para máquinas onde eles podem ser mais úteis;
- escalabilidade: para atuar em sistemas maiores, basta adicionar mais agentes e aumentar sua diversidade.

Recentemente, o COAST liberou uma implementação de um ambiente que segue as idéias do sistema proposto por Crosbie e Spafford. Este ambiente denominado *Autonomous Agents for Intrusion Detection* (AAFID) foi implementado utilizando-se a linguagem de *scripts* Perl e diversos recursos de administração de sistemas e segurança comuns em ambiente Unix. O ambiente AAFID possui duas entidades distintas que suportam a execução dos agentes do sistema: *Transceivers* e *Monitors*, sendo estes últimos entidades de mais alto nível, que podem detectar possíveis eventos intrusivos não notados por entidades mais simples como *Transceivers*. As informações preliminares sobre o sistema AAFID podem ser encontradas em (Zamboni et al, 1998).

Outro trabalho recente na área de aplicação de agentes autônomos em sistemas de detecção de intrusão é apresentado por Barrus e Rowe (Barrus & Rowe, 1998). A idéia dos autores é utilizar agentes autônomos estáticos que se comunicam através de um sistema de mensagens de alerta através de uma arquitetura distribuída. Algumas abordagens interessantes sugeridas são: a utilização de agentes especializados na detecção baseada em anomalia e na detecção baseada em uso indevido; e a criação de objetos específicos para tratar os diversos tipos de ataque.

3.4.5 O Sistema de Detecção de Intrusão do ICMC

Uma das características inovadoras deste sistema de detecção de intrusão (Cansian et al., 1997a, 1997b, 1997c; Bonifácio Jr. et al., 1998a) consiste em introduzir um “agente” de

segurança capaz de detectar comportamento intrusivo em conexões estabelecidas. Este agente atua capturando e decifrando pacotes que são transmitidos através da rede sob monitoramento. Para fazer uma inferência sobre a condição de segurança das conexões, o agente emprega um sistema especialista e uma rede neural que irá prover um coeficiente de suspeita, o qual, baseado em informações intrusivas previamente registradas, dará uma idéia a respeito da severidade do ataque ou o grau de suspeita das atividades naquela conexão.

O sistema se baseia no fato de que uma intrusão pode ser detectada a partir de uma análise de modelos predeterminados, que são anômalos se comparados com ações normais. A grande maioria dos ataques são resultado de um pequeno número de ataques conhecidos, como relatados por equipes como o CERT.

O uso de redes neurais pode fornecer mecanismos para o reconhecimento de ataques, bem como uma capacidade de adaptação em resposta a mudanças nas técnicas de intrusão. A Figura 5 mostra a arquitetura global do sistema.

Figura 5 Arquitetura global do sistema de detecção de intrusão do ICMC

3.5 Integração das Diversas Técnicas

A literatura tem mostrado que o uso integrado das diversas técnicas apresentadas neste capítulo é, certamente, o melhor método de prevenção e tratamento dos problemas de segurança. Como as tarefas executadas por cada uma das técnicas são muitas vezes complementares e poucas vezes redundantes, a troca de informações entre os sistemas (e.g., detecção de intrusão e *firewall*) permite uma atuação mais consistente e eficaz na determinação dos problemas de segurança e na solução, seja ela manual ou automática, dos mesmos.

Alguns exemplos de trabalhos nesta linha de pesquisa são apresentados por Bonifácio (Bonifácio Jr. et al., 1998b) e Mounji e Le Charlier (Mounji & Le Charlier, 1997). Outro sinal da tendência de integração dos sistemas de segurança é o estudo de um protocolo para troca de informações entre dispositivos e sistemas relacionados a segurança, denominado IDIP (*Intrusion Detection and Isolation Protocol*), que está sendo pesquisado e desenvolvido por parceiros como Boeing, *Trusted Information Systems* e *University of California at Davis* com financiamento do DARPA (DARPA, 1997).

3.6 Sumário

O crescimento no número de computadores conectados a Internet, bem como a grande disponibilidade de informações sobre segurança e quebra de segurança, têm aumentado enormemente o número de ataques aos sistemas de computadores. Tal situação pode provocar perdas enormes e há algum tempo diversas pesquisas vêm sendo realizadas para combater os problemas relacionados a segurança.

Técnicas clássicas como criptografia, autenticação e autorização têm sido utilizadas para garantir o bom funcionamento dos sistemas. No entanto, o simples uso destas técnicas não garante de forma alguma a segurança de uma rede de computadores. Por isso, técnicas complementares como o uso de *firewalls*, sistemas de análise de configuração e de detecção de intrusão são utilizados de forma que haja uma chance maior de evitar maiores comprometimentos. A utilização integrada dessas diversas técnicas também vem sendo pesquisada, permitindo a troca de informações entre os mesmos e, portanto, possibilitando uma abordagem holística da questão de segurança.

4 Gerenciamento de Segurança Apoiado por Agentes Móveis

Como foi descrito no Capítulo 3, novas abordagens estão sendo pesquisadas para a implementação de sistemas de gerenciamento de segurança. O enfoque até o momento tem sido na análise direta dos pacotes trafegando pela rede e na comparação com padrões estáticos previamente definidos. A inclusão de inteligência artificial, ou mais especificamente de sistemas baseados em regras, de redes neurais e de técnicas de análise e extração de informação de dados como *data mining* têm sido constante nos sistemas atuais.

Além disso, os sistemas que realizam a monitoração de segurança frequentemente também são alvos de ataques. Assim, é interessante que exista alguma forma de detecção de falhas nestes sistemas e de possibilidade de recuperação, seja por execução redundante em outras máquinas da rede, seja por reinício do processo do sistema de monitoração.

Desta forma, este trabalho sugere as seguintes diretrizes para a implementação de um ambiente que possa atuar nas condições citadas no parágrafo anterior:

- criação de um ambiente baseado em agentes móveis, com o intuito de prover as características de tolerância à falhas e mobilidade, além da possibilidade de configuração e instalação de agentes de segurança remotamente, facilidades para a definição de tarefas executadas pelos agentes, etc;
- definição de um *framework* e de uma interface de programação de aplicações¹⁰ que permitam estender as funcionalidades dos agentes, bem como incluir novos modelos de distribuição de objetos (e.g., CORBA ou RMI), novas linguagens de descrição de tarefas, etc.

¹⁰ Do inglês *API*, ou *Application Programming Interface*

- avaliar a possibilidade de integração do ambiente de agência em ferramentas de gerenciamento de redes, como o NetTracker.

As próximas seções deste capítulo apresentam uma visão geral do ambiente desejado, a especificação de requisitos e a análise do domínio do problema e o projeto das diversas partes do sistema.

4.1 Visão Geral

O objetivo desta seção é apresentar em termos gerais o que poderia ser considerado como um ambiente ideal para a execução de tarefas relativas ao gerenciamento de segurança de uma rede de computadores.

O gerenciamento de segurança depende de um processo contínuo de melhoramento e, portanto, de elaboração e avaliação das políticas de segurança, de renovação dos recursos de segurança e da correção de possíveis problemas. Assim, é possível imaginar o trabalho do gerente de segurança como a combinação harmônica de algumas tarefas básicas: **planejar, implementar, avaliar e corrigir** (Figura 6).

Figura 6 A Tarefa de Gerenciamento de Segurança

Certamente, a execução de todas estas tarefas demanda conhecimento amplo e dedicação do gerente de segurança. Ainda, dois fenômenos vêm ocorrendo paralelamente no decorrer da última década:

- o número de redes de computadores existentes está aumentando muito e, portanto, a demanda por profissionais capacitados para executar o gerenciamento de segurança também;
- informações sobre segurança estão disponíveis e organizadas na WWW, portanto, a complexidade da tarefa do gerente de segurança está ainda maior, pois mais pessoas tentam penetrar as diversas redes.

Considerando-se estes fenômenos, é ainda maior a urgência por sistemas que auxiliem a tarefa do gerente de segurança. Um aspecto natural destes novos sistemas deve ser a **delegação de tarefas**, ou seja, o gerente de segurança dispõe de meios para definir tarefas específicas e determinar que um componente do sistema o represente na realização da tarefa definida.

Portanto, a utilização de agentes de software pode fornecer algumas soluções interessantes para a implementação de um sistema de gerenciamento de segurança com as características de delegação citadas. Adicionalmente, o uso de agentes de software permite que o gerente de segurança defina regras, a partir das quais um agente pode tomar decisões independentemente.

Por outro lado, em caso de uma invasão bem sucedida de alguma máquina dentro do domínio gerenciado, faz-se necessária a existência de mecanismos de recuperação. Por exemplo, se um equipamento que funciona como roteador tem sua tabela de rotas alterada, toda a rede será prejudicada. Em casos como este, o uso de agentes móveis poderia facilitar a correção do problema, permitindo o retorno mais rápido às condições normais de funcionamento da rede.

4.1.1 Uma Metáfora para o Ambiente

Uma possível metáfora para o sistema de gerenciamento de segurança a ser desenvolvido foi inspirada na tradição medieval japonesa e incorpora as lendárias figuras dos Shoguns, ou senhores da guerra, e dos Samurais, guerreiros sem medo que seguiam um código de conduta denominado Bushido, ou “O Caminho do Guerreiro” (McGee, 1997).

Grosso modo, é possível fazer uma analogia da seguinte maneira. O Shogun seria um sistema gerente responsável por enviar novos agentes, os Samurais, doutrinados com o Bushido, que corresponde às tarefas de segurança definidas pelo administrador e que serviriam como direção inicial para os agentes de segurança.

A metáfora do Shogun e dos Samurais será adotada por ter sido considerada interessante em termos de segurança. Entretanto, outras metáforas similares existem e também podem ser consideradas para entender o funcionamento do sistema. Por exemplo, o sistema de defesa do organismo, no qual células especializadas realizam o trabalho de destruição de elementos considerados estranhos, ou sociedades de insetos como abelhas, formigas e cupins, nas quais indivíduos especializados realizam a tarefa de proteção dos seus membros e recursos, etc.

4.2 Especificação de Requisitos e Análise do Domínio

A especificação de requisitos é uma fase fundamental para a organização e desenvolvimento de um *software*. Entretanto, para que seja eficaz, um documento de requisitos precisa seguir algumas diretrizes básicas (Turine & Masiero, 1996):

- organizar os requisitos em torno de casos de uso do usuário (*use-case driven*);
- identificar unicamente cada requisito para posterior referência e
- separar requisitos funcionais (o que o sistema deve fazer) dos não funcionais (características de qualidade).

Portanto, o passo inicial é identificar os casos de uso do gerente de segurança. A partir disto, é possível ter uma visão mais precisa das funcionalidades que o ambiente deve realmente suportar, para que decisões mais explícitas possam ser tomadas durante as fases de projeto e implementação.

4.2.1 Documento de Requisitos

Nesta seção, serão apresentados os requisitos do sistema de acordo com as diretrizes já definidas. Em princípio, é possível identificar os seguintes casos de uso de alto nível do gerente

de segurança: manter lista de equipamentos gerenciados, gerenciar equipamentos remotamente e gerenciar agentes.

a. Manutenção da lista de equipamentos gerenciados

- a.1. sistema deve permitir a manutenção de uma lista de equipamentos gerenciados, ou seja, deve permitir as tradicionais operações de inclusão, remoção, alteração e consulta;
- a.2. sistema deve permitir a definição de certos atributos pertinentes aos equipamentos, mais especificamente, nome, endereço IP, arquitetura, localização física e outros atributos que o usuário possa desejar manter;
- a.3. sistema deve permitir a criação livre de grupos de equipamentos a partir da lista de equipamentos gerenciados;
- a.4. sistema deve permitir que todas as operações realizadas sobre equipamentos possam também ser efetuadas sobre grupos de equipamentos;
- a.5. sistema não deve permitir que usuários não autorizados tenham acesso à lista de equipamentos gerenciados;

b. Gerenciamento remoto dos equipamentos

- b.1. sistema deve permitir que o usuário associe agentes a equipamentos, ou grupos de equipamentos;
- b.2. sistema deve permitir que o usuário visualize, para cada equipamento gerenciado, uma lista dos eventos ocorridos;
- b.3. sistema deve permitir que o usuário visualize, para cada equipamento gerenciado, os resultados de execução de agentes;

c. Gerenciamento de agentes

- c.1. sistema deve permitir que o usuário faça o rastreamento dos agentes do sistema, ou seja, deve permitir que o usuário identifique quais agentes estão executando em quais equipamentos da rede;

- c.2. sistema deve permitir que o usuário tenha informações sobre o estado atual de execução de um agente, ou seja, se ele está ativo ou não;
- c.3. sistema deve permitir que o usuário altere o estado de execução de um agente, ou seja, ao usuário deve ser dado o direito de ativar ou interromper a execução de um agente;
- c.4. sistema deve permitir que o usuário defina um intervalo de tempo para execução das tarefas dos agentes
- c.5. sistema deve permitir a criação de um itinerário de execução para os agentes, ou seja, o sistema deve permitir que o usuário defina qual equipamento, ou grupo de equipamentos, será visitado por um agente;
- c.6. sistema deve permitir que o agente informe o usuário de eventos ocorridos, tais como, suspeitas de invasão, falhas de execução, etc.
- c.7. sistema deve permitir a criação de agentes através de uma linguagem de *scripts*;
- c.8. sistema deve permitir que o usuário faça composição de agentes predefinidos, criando novos agentes com o comportamento resultante da execução daqueles agentes;
- c.9. sistema deve permitir que o usuário associe recursos adicionais que possam ser necessários a execução de um agente, um exemplo de tal recurso seria o arquivo de uma biblioteca dependente de máquina que implementa captura de pacotes da rede;

d. Requisitos de Qualidade

- d.1. sistema deve ser de fácil manutenção, considerando-se que outras pessoas devem continuar o projeto;
- d.2. sistema deve ser portátil, considerando-se que as redes de computadores ainda são sistemas heterogêneos, ou seja, diversas arquiteturas e sistemas operacionais estão presentes;

- d.3. sistema deve apresentar facilidade de uso (características de usabilidade), incluindo-se neste critério: facilidade de instalação, presença de interface gráfica, possibilidade de programação visual e uso de *scripts*, etc;
- d.4. sistema deve possuir recursos avançados de segurança, pois trata-se de uma tarefa crítica para o funcionamento da rede: gerenciar a segurança da mesma.

4.2.2 Diagrama de Casos de Uso do Sistema

Um diagrama de casos de uso do sistema ajuda a documentar e a identificar os requisitos. A Figura 7 apresenta o diagrama para os casos de uso de alto nível para o ambiente.

Figura 7 Diagrama de Casos de Uso do Gerente de Segurança

4.2.3 Diagramas de Classes do Sistema

A Figura 8 apresenta um diagrama parcial com as classes principais do sistema, como identificadas a partir dos requisitos.

Figura 8 Diagrama de Classes do Ambiente Shogun

4.3 Projeto do Sistema

A partir desta seção, o ambiente de gerenciamento de segurança será referenciado como ambiente Shogun. O ambiente Shogun é composto por diversos módulos com diferentes níveis de complexidade. Esta seção procura documentar diversas faces do projeto do sistema.

4.4 Arquitetura do Ambiente

O ambiente possui dois tipos de elementos principais que implementam as funcionalidades especificadas:

- Monitor – o elemento monitor é um *host* no qual estão armazenadas as definições dos *hosts* gerenciados e dos agentes do sistema. A função do monitor é permitir que o

usuário configure seu ambiente de gerenciamento através da associação de agentes e *hosts*, bem como de despachar os agentes associados para os *hosts* remotos. Além disso, o monitor deve manter a auditoria das operações realizadas pelos agentes.

- Gerenciador – o elemento gerenciador é aquele responsável pela execução dos agentes de gerenciamento de segurança. Em cada *host* gerenciado deve estar instalado um *daemon* que receberá os agentes enviados por elementos do tipo Monitor que estejam certificados.

O elemento monitor deve permitir que o usuário do ambiente tenha acesso às suas funções através de uma aplicação de gerenciamento de segurança. A Figura 9 apresenta a arquitetura do ambiente através das relações entre os diversos elementos e aplicações.

Figura 9 Arquitetura do Ambiente Shogun

Os itens seguintes fornecem algumas informações sob diferentes partes do ambiente proposto.

4.4.1 Modelo de Distribuição de Objetos

Para realizar a comunicação entre os componentes do ambiente Shogun que executam em *hosts* distintos da rede deverá ser utilizado o padrão RMI (*Remote Method Invocation*). A decisão pela utilização desta tecnologia de distribuição de objetos e não de CORBA, advém da maior integração de RMI com a linguagem Java e, portanto, da maior facilidade para o desenvolvimento de um protótipo inicial.

4.4.2 Definição de uma Linguagem de Script

Para permitir a fácil extensão do ambiente e a criação de módulos-tarefa para os agentes, optou-se pela utilização de uma linguagem de *scripts*. Dentre as linguagens deste tipo que são amplamente utilizadas, é possível citar: Perl, TCL e Python, entre outras.

Perl é uma linguagem que possui facilidades para o processamento de cadeias de caracteres e geração de relatórios. Por isso, vem sendo muito utilizada com linguagem de programação de aplicações de administração de sistemas (AAFID, p.e.)

Considerando-se o nível de coesão, de simplicidade da sintaxe e, principalmente, de integração com a linguagem Java, Python foi a linguagem escolhida. Diferentemente das outras linguagens citadas, Python não necessita de código dependente de arquitetura para ser utilizado, pois possui uma versão de interpretador feita totalmente em Java. Além disso, permite criação de componentes Java (*Java Beans*), compilação dos *scripts* para bytecodes Java e execução do interpretador a partir de um objeto Java, entre outras características.

Entretanto, apesar de Python ter sido escolhida como linguagem de *script* para o protótipo inicial, o sistema deve prever, de acordo com os requisitos, a utilização de outras linguagens.

4.4.3 Características de Segurança

Como anteriormente observado no Capítulo 2, agentes podem ser facilmente caracterizados como vírus. Portanto, é fundamental que o sistema faça uso intenso de recursos de segurança para prevenir a execução de comportamento não adequado.

No ambiente Shogun, os recursos de segurança oferecidos pela API 1.2 da linguagem Java foram amplamente utilizados para manter a segurança dos agentes enquanto estes trafegam pela rede e para proteger os seus *hosts* de possíveis ataques advindos de agentes não autorizados.

Um dos objetos-chave para implementação de restrições de segurança dentro da máquina Java é o *SecurityManager*. Um *SecurityManager* implementa controle fino sobre as diversas operações que podem ser realizadas dentro do ambiente Java. Este controle abrange desde operações de escrita e leitura a dispositivos do sistema operacional até operações de mais alto nível como a possibilidade de fazer inspeção em objetos através da *Reflection API*. Assim, uma das

características do ambiente Shogun é que ele possui uma implementação própria da classe *SecurityManager* para definir as restrições de acesso que se aplicam aos agentes que chegam ao sistema.

Outro recurso interessante presente na API 1.2 é a possibilidade de definição de políticas específicas para determinados objetos, pacotes, certificadores, etc. Através do uso desta funcionalidade, um sistema pode restringir o acesso a um nível seguro através do *SecurityManager* e permitir que objetos internos tenham maior liberdade na execução de rotinas críticas.

4.4.4 Ambiente de Agência

Como especificado pelo documento de requisitos, o ambiente Shogun faz uso de agentes móveis para realizar as tarefas de gerenciamento de segurança. Contudo, seria interessante possuir um ambiente de agência independente da aplicação de gerenciamento, mas que fornecesse as características necessárias de segurança, performance, etc, necessárias a execução de tal aplicação. Para tanto, é necessário que o ambiente de agência disponibilize alguns serviços aos agentes, entre eles: criação de agentes, definição de tarefas, comunicação entre agentes, transferência dos agentes entre ambientes de agência, transferência dos recursos associados e segurança do sistema.

4.4.4.1 Criação dos Agentes

Um problema a respeito da criação de agentes é que, em muitos ambientes de agência existentes, não existe nenhum controle a respeito de quais objetos podem ser agentes e utilizar os recursos de migração do ambiente, por exemplo. Para resolver esta questão, é possível exigir que objetos que servirão como containeres para agentes devam ser criados pelo próprio ambiente de agência. Desta forma, o ambiente de agência pode verificar se um agente, que está tentando executar uma operação, é válido ou não.

A comunidade que trabalha com *design patterns* denomina este tipo de construção *Creator Pattern* (Larman, 1997).

4.4.4.2 Definição de Tarefas ou Ações

Para implementar a funcionalidade de definição de tarefas, o uso de um *pattern* do tipo *Command* é interessante, pois através do uso do mesmo é possível sequenciar comandos, desfazer comandos e executar *logging*.

As tarefas que poderão ser executadas pelos agentes deverão ser registradas no ambiente, de modo que possa existir uma forma de controle sobre as mesmas. Tarefas definidas pelo usuário são implementadas através da interface *IAgentTask*.

A interface *IAgentTask* possui um método denominado *execute* que será invocado pelo ambiente de agência. Opcionalmente, a tarefa pode retornar um objeto qualquer como forma de fornecer valores de saída da sua execução.

Uma implementação pré-existente no sistema da interface *IAgentTask* e que dá suporte ao sequenciamento de múltiplas tarefas é a *AgentSequencedTasks*. A implementação desta classe recebe como entrada um vetor de tarefas predefinidas e as executa uma a uma. O retorno da execução do método *execute* é um vetor com os objetos retornados por cada uma das tarefas.

4.4.4.3 Transferência dos Agentes entre Ambientes de Agência

A transferência dos agentes entre ambientes de agência é suportada através do método *moveTo* do ambiente de agência.

Como resultado da invocação daquele método, o agente é enviado juntamente com seu estado atual para o *host* remoto. Deve-se notar que a operação é realizada com sucesso apenas se o *host* remoto executar os procedimentos de autorização e autenticação do agente enviado corretamente.

O agente pode, ainda, invocar o método *moveTo* independentemente do *host* em que ele está atualmente, mas qualquer chamada a este método fora do ambiente de agência no qual o agente foi criado, faz com que este seja retornado ao seu ambiente de origem. Esta operação é realizada de forma que seja possível realizar auditoria das migrações feitas pelo agente.

Um recurso interessante fornecido pelo ambiente é a possibilidade de criação e associação de itinerários (semelhantes aos usados pelos *Aglets* da IBM). Os itinerários referem-se a *hosts* consecutivos que um agente deve visitar. Entretanto, as migrações necessárias para execução do itinerário estão sujeitas às mesmas restrições impostas ao método *moveTo*.

Alternativamente, o agente pode ser clonado em um *host* remoto, ou seja, o agente é enviado para aquele *host*, mas a instância local do agente permanece intacta. Esta operação é realizada através da invocação do método *cloneAt* do ambiente de agência.

4.4.4.4 Transferência dos Recursos Associados

Como resultado da migração de um agente, pode ser necessária a transferência de recursos específicos, como bibliotecas de código nativo, ou arquivos de dados.

A transferência destes recursos é feita no sistema através de um protocolo de transferência de arquivos.

Para que um recurso possa ser transferido, ele deve estar registrado no sistema a semelhança de tarefas e agentes. Desta forma, arquivos que não possuem permissão explícita para transferência não podem ser requisitados por sistemas externos.

4.4.4.5 Comunicação entre Agentes

Como observado no Capítulo 2, muitos autores consideram que a capacidade de comunicação entre agentes é fundamental para *softwares* deste tipo.

A implementação atual do ambiente deve considerar uma forma de troca de mensagens entre os agentes semelhante a proposta pela arquitetura CORBA e pelo RMI, ou seja, chamada de métodos remotos.

Para tanto, cada agente pode ter associada uma lista com os nomes dos métodos publicados pelo agente. Quando um agente tenta enviar uma mensagem para outro, o ambiente de execução verifica se o método requisitado para a passagem da mensagem está publicado ou não.

Em implementações futuras, o suporte a uma linguagem de manipulação de conhecimento como KQML, ou mesmo ACL, deveria ser considerado.

4.4.4.6 Segurança do Sistema

Em se tratando de um ambiente que deverá ser utilizado para executar tarefas de gerenciamento de segurança, é ainda mais importante projetar um sistema que apresente características de segurança avançadas. Desta forma, alguns recursos para autenticar os agentes que se apresentam a um sistema receptor são: uso de certificados X.509, restrição por origem do agente, etc.

4.4.5 Serviço de Persistência

Qualquer aplicação que necessite manter intactos os dados para sua execução entre execuções distintas necessita de algum esquema de persistência, ou seja, armazenamento de estado em um dispositivo de armazenamento não volátil.

Entre os sistemas que podem implementar um serviço de persistência, estão desde arquivos texto até gerenciadores de bancos de dados relacionais e bases de dados orientadas a objetos.

No ambiente Shogun, o interesse por um serviço de persistência está em armazenar os objetos gerenciados pelo sistema, ou seja, *hosts* e agentes. *Hosts* devem ser armazenados de forma persistente simplesmente para que o sistema saiba quais *hosts* devem ser monitorados. Por outro lado, agentes precisam ser persistentes por motivos mais críticos ao funcionamento do ambiente, por exemplo: um agente pode estar coletando e analisando informações em um *host* específico e pode ter sua operação interrompida por uma queda de energia.

Em termos práticos, o ambiente Shogun definirá apenas a interface de acesso que o serviço de persistência deve oferecer. Durante diversos pontos da execução do ambiente, um par (identificador, objeto) será fornecido ao serviço de persistência, que deve armazenar o objeto utilizando o identificador como chave de acesso para futura recuperação.

A Figura 10 apresenta um diagrama de seqüência com as interações entre o ambiente Shogun e o serviço de persistência.

Figura 10 Diagrama de Seqüência (Interação entre o Ambiente Shogun e o Serviço de Persistência)

Um aspecto muito importante que deve ser considerado em relação ao serviço de persistência é a necessidade de controle de concorrência. Na implementação, os métodos desta classe precisam ser sincronizados através do uso de variáveis de condição ou de monitores, que são os recursos de sincronização disponíveis na linguagem Java.

4.4.6 Aplicação de Gerenciamento

A aplicação de gerenciamento constitui-se na interface gráfica de gerenciamento que faz acesso remoto aos serviços de gerenciamento do módulo Monitor.

Para que o usuário possa utilizar o ambiente, é necessário que ele execute um *login* no módulo monitor. Uma vez que esta operação tenha sido efetuada com sucesso e o usuário esteja autenticado com o Monitor, será possível utilizar os recursos de gerenciamento de *hosts* e agentes através de uma interface gráfica.

Deve-se notar que, se decorrer um intervalo de tempo determinado com ausência de operações entre a aplicação de gerenciamento e o Monitor, a autenticação do usuário é revogada. Desta

forma, evita-se que outros usuários utilizem dos recursos da aplicação em situações que o usuário legítimo esquece-se de terminar a execução da interface gráfica.

As Figuras abaixo (Figura 11 e Figura 12) apresentam a tela inicial de Login e a interface de gerenciamento básica do sistema.

Figura 11 Tela de Login do Ambiente Shogun

Figura 12 Tela da aplicação de gerenciamento do Ambiente Shogun

4.4.6.1 O editor de propriedades

No ambiente Shogun, agentes são implementados e tratados também como JavaBeans™. Desta forma, agentes especializados criados pelos usuários do sistema podem conter atributos específicos às suas tarefas, permitindo que se possa criar um interface do usuário baseado no uso de componentes de *software*.

Para que o sistema apresente esta facilidade, foi implementado um editor de propriedades semelhante ao existente em ambientes de programação visual como Visual Basic e Delphi. Entretanto, para que o editor de propriedades possa ser utilizado é necessário que os agentes implementados obedeçam as regras de nomenclatura de métodos fornecidas pela especificação JavaBeans da Sun.

4.4.7 Organização da Arquitetura em Termos de Pacotes

A arquitetura do ambiente Shogun apresentada no item anterior pode ser visualizada como uma série de pacotes de *software* com várias dependências. A Figura 13 apresenta um diagrama de pacotes UML, com todos os pacotes que implementam o ambiente.

Figura 13 Diagrama de Pacotes para o Ambiente Shogun

4.5 Um Exemplo de Agente de Segurança

Um agente de segurança que pode ser utilizado na prática seria um verificador de arquivo de senhas para Unix. Para uma tarefa como esta, o agente atua como um sistema de análise de configuração, procurando por possíveis falhas de configuração. Alguns problemas que podem ser

detectados são: a existência de mais de uma entrada com direitos de super-usuário, nomes de usuários que não são formados apenas por letras e dígitos, etc.

Os passos para a implementação e integração de um agente deste tipo ao ambiente são:

- Criação da tarefa – este passo pode ser executado pelo gerente do sistema de duas formas distintas, através da implementação da interface *IAgentTask*, ou através da criação de uma tarefa baseada em *script* Python;
- Registro da tarefa no ambiente Shogun – para ser utilizada dentro do ambiente a classe precisa registrar-se com o mesmo. Este passo acrescenta as classes e recursos necessários para execução de uma instância do agente à biblioteca de classes do gerenciador;
- Associação do agente com *hosts* gerenciados – este passo consiste na utilização efetiva do agente no ambiente. Uma vez associado a um ou mais *hosts*, o agente será despachado para estes e executará suas tarefas de gerenciamento de segurança.

4.6 Integração do Ambiente Shogun ao NetTracker

NetTracker é uma arquitetura operacional extensível para ferramentas de gerenciamento de redes (Mouro, 1997). Este sistema possibilita a criação de aplicações de gerenciamento e a integração desta a um ambiente que permite entre outras coisas: gerenciamento da aplicação através da WWW, carga e descarga dinâmica de módulos, configuração de parâmetros da aplicação e definição de conteúdo HTML como saída da aplicação. Alguns detalhes sobre o funcionamento e arquitetura do NetTracker são apresentados no Apêndice A.

Uma das metas deste trabalho foi avaliar a possibilidade de integração do ambiente Shogun ao NetTracker. Esta possibilidade revelou-se plausível e torna-se útil para o gerenciamento de elementos monitores do ambiente Shogun através do NetTracker.

Esta integração foi possível devido à separação entre interface do usuário e módulos funcionais no ambiente Shogun. Desta forma, foi possível integrar os ambientes em uma nova arquitetura demonstrada na Figura 14.

Figura 14 Integração do Ambiente Shogun ao NetTracker

Entretanto, o protótipo atual considera apenas a possibilidade de carregar e descarregar a aplicação que suporta o módulo monitor do ambiente Shogun e a capacidade de fornecer uma interface de gerenciamento através de *browser*. O código para a aplicação que integra os módulos, suportando as características referidas, está apresentado na Figura 15.

Figura 15 Código de Integração do Ambiente Shogun ao NetTracker

```

import shogun.environment.*;
import java.io.*;

public class ShogunManagementApplication extends MVApplication {
    public ShogunManagementApplication() {
        super("Shogun Security Management Application");
    }

    // cria a instancia do elemento Monitor e inicia sua execucao
    public void run() {
        ShogunMonitor sm = new ShogunMonitor();
        sm.start();

        this.thread.suspend();
    }

    public Reply perform( Request request ) {
        request.makeReply(200);

        // Fornece a URL do Applet de Gerenciamento
        PrintStream p = new PrintStream(request.getOutputStream());
        p.println(ShogunUI.getAppletTag());
        // Retorna a resposta de maneira correta
        return new Reply(p);
    }
}
  
```

Caso o usuário opte por carregar o módulo de gerenciamento de segurança sempre que o NetTracker for inicializado, é possível adicionar uma entrada com o nome do módulo ao arquivo MVAppReg.cfg. Deve-se notar que a integração entre os dois ambientes pode ser ainda melhorada, pois o recurso de armazenamento de parâmetros (atributos) da aplicação não foi explorada

4.7 Sumário

O objetivo deste capítulo foi apresentar os requisitos e a arquitetura proposta para o ambiente de gerenciamento de segurança.

Como características importantes, pode-se destacar:

- a “elicitación” dos requisitos necessários a implementação de um ambiente de gerenciamento de segurança
- a implementação de um ambiente de agência compatível com os requisitos do domínio de aplicação de segurança;

5 Conclusão e Trabalhos Futuros

O objetivo deste trabalho foi especificar os requisitos, determinar uma arquitetura plausível e fornecer a implementação de um protótipo de um ambiente de gerenciamento de segurança apoiado por agente móveis.

O ambiente projetado e prototipado pode ser considerado um avanço importante, pois, além de fornecer recursos tecnológicos avançados e consoantes com os desenvolvimentos da área, permite uma abordagem holística do problema do gerenciamento de segurança.

Considerando-se as características de segurança apresentadas, é possível dizer que o ambiente Shogun é relativamente seguro e, portanto, é possível utilizá-lo como ferramenta para o gerenciamento de segurança. No entanto, restam ainda os problemas relativos ao ataque de *hosts* mal intencionados ao ambiente de agência e aos agentes, porém, esta questão ainda é tópico de pesquisa intensa na comunidade internacional e, provavelmente, será tratada em implementações posteriores do ambiente Shogun.

5.1 Avaliação

O protótipo do ambiente implementado foi testado através de operações práticas, porém, há a necessidade de uma avaliação mais criteriosa e, possivelmente, de testes formais.

Alguns pontos que devem ser considerados durante a avaliação do ambiente em trabalhos futuros são:

- desempenho em termos de uso de memória, *overhead* no processamento das máquinas gerenciadas, transmissão pela rede, etc. Com relação a esta tipo de medição, estão disponíveis ferramentas como o *profiler* da linguagem Java, bem como a sua API.

- adequação à tarefa de gerenciamento de segurança. Este ponto é crucial para o sucesso do ambiente e da arquitetura propostos, entretanto, somente o uso real e por longo tempo de uma versão mais aprimorada dos módulos do ambiente poderá comprovar com certeza a adequação dos mesmos.

5.2 Trabalhos Futuros

Além de uma avaliação mais precisa das funcionalidades oferecidas pelo ambiente, pode-se destacar como possíveis trabalhos futuros:

- utilização de CORBA/IIOP como arquitetura de distribuição de objetos, bem como de serviços como a MAF para suportar agentes móveis;
- utilização de arquiteturas voltadas para domínios de aplicação específicos, que possam ser eventualmente apresentadas pela FIPA;
- construção de adaptadores para outras linguagens de *script* menos integradas com Java, como Perl e TCL, mas com as quais a grande maioria dos especialistas de segurança possuem experiência;
- realizar esforços com o intuito de tentar quebrar propositadamente as restrições de segurança impostas pelo ambiente. Estes esforços serviriam para validar os recursos de segurança apresentados pelo ambiente; e
- inclusão de funcionalidades que permitam a utilização de técnicas de IA, como sistemas baseados em conhecimento, pelos agentes. Esta adição poderia fornecer grande flexibilidade aos agentes, transformando-os em agentes móveis inteligentes.

Referências Bibliográficas

- (Baldwin 1994) BALDWIN, R.W. Kuang: Rule-based Security Checking. MIT, Lab for Computer Science Programming Systems Research Group, 1994 (Relatório Técnico).
- (Barrus & Rowe 1998) BARRUS, J.; ROWE, N.C. A Distributed Autonomus-Agent Network-Intrusion Detection and Response System. In: Proceedings of the 1998 Command and Control Research and Technology. Monterrey CA, Junho-Julho 1998.
- (Bellovin & Cheswick 1994) BELLOVIN, S.; CHESWICK, W. Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley. 1994.
- (Bonifácio Jr. et al. 1998a) BONIFÁCIO Jr., J. M.; CANSIAN, A.M.; CARVALHO, A.C.P.L.; MOREIRA, E.S. Neural Networks Applied in Intrusion Detection Systems. In: *Proceedings of the IEEE IJCNN '98 International Joint Conference on Neural Networks*, Anchorage, Alaska, Maio 1998.
- (Bonifácio Jr. et al. 1998b) BONIFÁCIO Jr., J. M.; CANSIAN, A.M.; CARVALHO, A.C.P.L.; MOREIRA, E.S. *Um Ambiente de Segurança Distribuído para a Integração de Firewalls com Sistemas de Detecção de Intrusão*. In: XVI Brazilian Symposium on Computer Networks, SBRC'98. Rio de Janeiro, 1998.
- (Cansian et al. 1997a) CANSIAN, A.M.; MOREIRA, E.M.; CARVALHO, A.C.P.L.; BONIFÁCIO Jr., J.M. Network Intrusion Detection Using Neural Networks. In: *Proceedings of International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'97*, Gold Coast, Australia, p.276-280, Fevereiro 1997.
- (Cansian et al. 1997b) CANSIAN, A.M.; MOREIRA, E.M.; MOURO, R.B.; MORISHITA, F.T.; CARVALHO, A.C.P.L.; An Adaptative System for Detecting Intrusion in Networks. In: *Proceedings of the III International Congress on Information Engineering*, Buenos Aires, Argentina, p.96-105, Abril 1997.
- (Cansian et al. 1997c) CANSIAN, A.M.; MOREIRA, E.M.; CARVALHO, A.C.P.L.; BONIFÁCIO Jr., J.M. Um Modelo Adaptativo para Detecção de Comportamento Suspeito em Redes de Computadores. In: *Proceedings of the XV Brazilian Symposium on Computer Networks, SBRC'97*, p. 51-60, São Carlos, Maio 1997.

- (Cansian 1997) CANSIAN, A. M. *Desenvolvimento de um sistema adaptativo de detecção de intrusos em redes de computadores*. São Carlos, 1997. 153p. Tese (Doutorado) – Instituto de Física de São Carlos, Universidade de São Paulo.
- (Chapman 1992) CHAPMAN, D. B. Network (In)Security Through IP Packet Filtering. In: *Proceedings of USENIX Security Symposium III*, p.63-76. Setembro 1992. WWW: <http://www.cert.lu/security/documents.html>
- (Chess et al. 1995) CHESS, D.; GROSOFF, B.; HARRISON, C.; LEVINE, D.; PARRIS, C. Itinerant Agents for Mobile Computing. IBM, Março 1995. (Relatório Técnico RC20010).
- (Cicilini 1994) CICILINI, R. *Desenvolvimento de um Agente SNMP para Plataformas Rodando DOS*. São Carlos, 1994. 107p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
- (Corley et al. 1998) CORLEY, S.; TESSELAAR, M.; COOLEY, J.; MEINKHN, J.; MALABOCCHIA, F.; GARJO, F. *The Application of Intelligent and Mobile Agents to Network and Service Management*. Fifth International Conference On Intelligence in Services and Networks. Antuerpia, Bélgica, Maio 1998. WWW: <http://www.alcatel.be/telecom/news/isn98>
- (Crosbie & Spafford 1995a) CROSBIE, M; SPAFFORD, E.H. Defending a Computer System using Autonomous Agents. Department of Computer Sciences, Purdue University, 1995. (Relatório Técnico CSD-TR-95-022; Coast TR 95-02). WWW: <http://www.cs.purdue.edu/homes/spaf/tech-reps/9522.ps>
- (Crosbie & Spafford 1995b) CROSBIE, M; SPAFFORD, E.H. Active Defense of a Computer System using Autonomous Agents. Department of Computer Sciences, Purdue University, 1995. (Relatório Técnico CSD-TR-95-008). WWW: <http://www.cs.purdue.edu/homes/spaf/tech-reps/9508.ps>
- (DARPA 1997) DEFENSE ADVANCED RESEARCH PROJECTS AGENCY. Adaptive Systems Security Policies. 1997. WWW: http://www.darpa.mil/ito/Summaries97/F256_0.html
- (Erickson 1997) ERICKSON, T. Designing As If People Mattered. In: BRADSHAW, J.M., ed. *Software Agents*. AAAI Press/MIT Press, 1997. Cap.5, p.79-96.
- (Farmer & Spafford 1990) FARMER, D.; SPAFFORD, E.H. COPS Security Checker System. In: PROCEEDINGS OF THE SUMMER USENIX

- CONFERENCE, Anaheim, CA. 1990. p.165-170. WWW: <ftp://coast.cs.purdue.edu/pub/Purdue/papers/spafford/farmer-spaf-cops.ps.Z>
- (Finin, Labrou & Mayfield, 1997) FININ, T.; LABROU, Y.; MAYFIELD, J. KQML as an Agent Communication Language. . In: BRADSHAW, J.M., ed. *Software Agents*. AAAI Press/MIT Press, 1997. Cap. 14, p.291-316.
- (FIPA 1997) FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS; Agent Communication Language. In: FIPA'97 Specification Part 2. 1997. WWW: <http://drogo.cselt.stet.it/ufv/leonardo/fipa/spec/fipa7112.zip>
- (Franklin & Graesser 1996) FRANKLIN, S.; GRAESSER, A. Is It An Agent, or Just a Program? A Taxonomy for Autonomous Agents. In: *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag. 1996. WWW: Agent or Program, <http://www.mscl.memphis.edu/~franklin/AgentProg.html>
- (Fuggeta, Picco & Vigna 1998) FUGGETA, A.; PICCO, G.P.; VIGNA, G. Understanding Code Mobility. In: IEEE Transactions on Software Engineering, vol. 24, no. 5, p.342-361. Maio 1998.
- (Genesereth & Ketchpel 1994) GENESERETH, M. R.; KETCHPEL, S. P. Software Agents. In: Communications of the ACM, vol. 37, no. 7, p.48-53. 1994.
- (Genesereth & Fikes 1992) GENESERETH, M. R.; FIKES, R. E. Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University. March. 1992. (Technical Report Logic-92-1). WWW: <http://www-ksl.stanford.edu/knowledge-sharing/kif/>
- (Gruber 1993a) GRUBER, T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: GUARINO, N.; POLI, R., ed. *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers. 1993.
- (Gruber 1993b) GRUBER, T.R. A Translation Approach to Portable Ontology Specifications. In: *Knowledge Acquisition*, no. 5, vol. 2, p.199-220, 1993. WWW: <ftp://ksl.stanford.edu/pub/knowledge-sharing/papers/>.
- (Harrison, Chess & Kershbaum 1997) HARRISON, C.G.; CHESS, D.M.; KERSHENBAUN, A. Mobile Agents: Are they a good idea? In: *Mobile Object Systems: Towards the Programmable Internet*. (ed) VITEK, J.; TSCHUDIN, C. Lecture Notes in Computer Sciences, vol. 1222. p.25-47. Springer Verlag. 1997. Disponível também como

- Relatório Técnico da IBM. WWW: http://www.research.ibm.com/iagents/paps/mobile_idea.ps. 1994.
- (Heydon 1992) HEYDON, A. Specifying and Checking Unix Security Constraints. In: PROCEEDINGS OF THE 3rd USENIX SECURITY SYMPOSIUM. 1992
- (ISS 1995) INTERNET SECURITY SYSTEMS. Internet Scanner. 1995. (Relatório Técnico). WWW: <http://www.iss.net/iss/scanner.html>
- (Javitz et al., 1993) JAVITZ, H.S.; VALDES, A.; LUNT, T.F.; TAMARU, A.; TYSON, M.; LOWRANCE, J. Next Generation Intrusion Detection Expert System ({NIDES}). 1993. (Relatório Técnico SRI-A016-Rationales).
- (Jennings & Wooldridge 1995) JENNINGS, N.R.; WOOLDRIDGE, M. Intelligent Agents: Theory and Practice. In: Knowledge Engineering Review, vol. 10, no. 2, Junho 1995.
- (Kim & Spafford 1994) KIM, G.; SPAFFORD, E. Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection. In: PROCEEDINGS OF THE SYSTEMS ADMINISTRATION, NETWORKING AND SECURITY CONFERENCE III (SANS), Washington, DC. 1994. WWW: <ftp://coast.cs.purdue.edu/pub/COAST/papers/Tripwire.ps.Z>
- (Larman, 1997) LARMAN, C.; *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design*. Prentice Hall.
- (Lee & Stolfo 1997) LEE, W.; STOLFO, S. Data Mining Approaches for Intrusion Detection. In: *Proceedings 1998 7th USENIX Security Symposium*. 1997. WWW: <http://www.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html>
- (Lieira 1995) LIEIRA, J. F. *Utilização de Áudio e Vídeo em Sistemas Gerenciadores de Redes de Computadores*. São Carlos, 1995. 111p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
- (Lunt & Jagannathan, 1988) LUNT, T.F.; JAGANNATHAN, R. A Prototype Real-Time Intrusion Detection Expert System. In: *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, Abril 1988.
- (McGee, 1997) MCGEE, M. A Brief History of the Samurai. 1997. WWW: http://www.mindspring.com/~mamcgee/iaido_samurai.html
- (Milojovic, Guday & MILOJICIC, D. S., GUDAY, S., WHEELER, R. *Old Wine in New Bottles - Applying OS Process Migration Technology to Mobile*

- Wheeler 1997) *Agents*. In: Mobile Object Systems ECOOP Workshop'97. Jyväskylä, Finland, June 9-10, 1997. WWW: <http://www.osf.org/RI/java/moa/Finland.fr.ps>
- (Moraes 1995) MORAES, S. *Voz em Sistemas Computacionais: Projeto e Implementação de Módulos de Processamento de Voz em Gerenciamento de Redes*. São Carlos, 1995. 103p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
- (Moreira & Walczowski 1997) MOREIRA, D.A.; WALCZOWSKI, L.T. Using Software Agents to Generate VLSI Layouts. IEEE Expert. p26-32. November-December 1997.
- (Morishita 1997) MORISHITA, F. T. *Uma Avaliação Evolutiva dos Protocolos de Gerenciamento da Internet: SNMPv1, SNMPv2 e SNMPv3*. São Carlos, 1997. 68p. Dissertação (Mestrado) – Instituto de Ciência Matemáticas e de Computação, Universidade de São Paulo.
- (Mounji & Le Charlier 1997) MOUNJI, A.; LE CHARLIER, B. Continuous Assesment of a Unix Configuration: Integrating Intrusion Detection and Configuration Analysis. In: Proceedings of the IEEE ISOC'97 Symposium on Network and Distributed Systems Security. 1997. WWW: <http://www.info.fundp.ac.be/~amo/publication.html>
- (Mouro 1997) MOURO, R. B. *Uma Arquitetura Operacional Extensível para Ferramentas de Gerenciamento de Redes*. São Carlos, 1997. 60p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
- (Mouro, Morishita & Moreira 1997) MOURO, R. B.; MORISHITA, F. T.; MOREIRA, E. S. *NetTracker :uma arquitetura operacional extensível para ferramentas de gerenciamento de redes*. São Carlos, 1997. 15º Simpósio Brasileiro de Redes de Computadores. p164-174. 1997.
- (Nicolas 1998) NICOLAS, P.R. Agent-based Workflow Automation. In: WWW: White Paper: Agent-based Workflow Automation, <http://www.ikonodyne.com/whitewkflow/agents.html>. Ikonodyne Inc. 1998.
- (Nwana & Wooldridge 1996) Nwana, H.; WOOLDRIDGE, M. Software Agents Technologies. In: BT Technology Journal, vol. 14, no. 4, p68-78. Outubro 1996. WWW: http://www.labs.bt.com/projects/agents/publish/papers/sat_report.htm
- (Nwana 1997) Nwana, H.S. Software Agents: An Overview. In: Knowledge Engineering Review, vol. 11, no. 3, p205-244,

- (Oda 1994) ODA, C. S. *Desenvolvimento de um Sistema Monitor Gráfico Baseado em Protocolo de Gerenciamento SNMP*. São Carlos, 1994. 111p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
- (Porra & Neumann 1996) PORRAS, P.A.; NEUMANN, P.G. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances – Conceptual Overview. Dezembro 1996. (Relatório Técnico). WWW: <ftp://ftp.csl.sri.com/pub/emerald-position1.ps>
- (Porras & Neumann 1997) PORRAS, P.A.; NEUMANN, P.G. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In: 20th National Information Systems Security Conference Proceedings, p.353-366. Baltimore, Maryland. Outubro 1997. WWW: <http://www.csl.sri.com/emerald/emerald-niss97.html>
- (Sander & Tschudin 1998a) SANDER, T.; TSCHUDIN, C.F. Protecting Mobile Agents Against Malicious Hosts. In: VIGNA, G. (ed.) *Mobile Agents and Security*. Lecture Notes in Computer Science, vol. 1419. Springer-Verlag. 1998.
- (Sander & Tschudin 1998b) SANDER, T.; TSCHUDIN, C.F. Towards Mobile Cryptography. In: *Proceedings of IEEE Security & Privacy '98*. Oakland, California, Maio 1998. WWW: <http://www.icsi.berkeley.edu/~sander/publications/satschu.ps>
- (Turine & Masiero, 1996) TURINE, M.A.S.; MASIERO, P.C. Especificação de Requisitos: uma Introdução. São Carlos, ICMSC, 1996. (Relatório Técnico ICMSC-USP, 39)
- (Venema 1995) VENEMA, W. Project Satan: Unix/Internet Security. In: PROCEEDINGS OF THE COMPSEC'95 CONFERENCE, Elsevier, London. 1995. WWW: <http://www.fish.com/zen/satan/satan.html>
- (Vitek, Serrano & Thanos 1997) VITEK, J.; SERRANO, M.; THANOS, D.. Security and Communications in Mobile Object Systems. In: *Mobile Object Systems: Towards the Programmable Internet*. (ed) VITEK, J.; TSCHUDIN, C. Lecture Notes in Computer Sciences, vol. 1222. Springer Verlag. 1997. WWW: <http://cuiwww.unige.ch/OSG/people/jvitek/Publications/tpi.ps.gz>
- (Vitek & Tschudin 1997) VITEK, J.; TSCHUDIN, C. (ed) *Mobile Objects Systems: Towards the Programmable Internet*. Lecture Notes in Computer Sciences, vol. 1222. Springer Verlag. 1997.
- (Wack & Carnahan) WACK, J. P.; CARNAHAN, L. Keeping Your Site Comfortably

1994)

Secure: An Introduction to Internet Firewalls. In: National Institute of Standards and Technology Publication. 1994. WWW: <http://www.raptor.com/lib/index.html>

(Zamboni et al.,
1998)

ZAMBONI, D.; SPAFFORD, E.; ISACOFF, D.; GARCIA-FERNANDEZ, J.O.; BALASUBRAMANIYAN, J.S. An Architecture for Intrusion Detection using Autonomous Agents. 1998. (Relatório Técnico COAST 98/05)

(Zerkle & Levitt
1996)

ZERCKLE, D.; LEVITT, K. NetKuang - A Multi-Host Configuration Vulnerability Checker. In: 6th USENIX SECURITY SYMPOSIUM, San Jose, CA. 1996. WWW: <http://seclab.cs.ucdavis.edu/papers/zl96.ps>

o Ambiente NetTracker

O NetTracker é um sistema de gerenciamento de redes que tem como característica principal o uso de recursos da WWW para seu acesso. Desta forma, é possível ativar módulos de gerenciamento remotamente e observar os resultados da execução destes módulos através de páginas HTML normais.

A estrutura do gerenciador (Mouro, 1997; Mouro, Morishita & Moreira, 1997) é a mais modular possível, de forma que partes possam ser inseridas e ou removidas sem a necessidade de grandes alterações no sistema, facilitando a manutenção do código fonte.

Figura 16 Relacionamento de comunicação intramódulos

A Figura 16 sintetiza o relacionamento de comunicação entre os três principais elementos que compõem o ambiente gerenciado pela ferramenta NetTracker.

Manager Resources são os elementos que compõem a rede e que sejam dotados de capacidades de gerenciamento. *Manager Station* é o centralizador das informações, fornece as funções de controle e monitoramento ao administrador. *Manager Interface* é a interface do sistema, disponibilizada ao administrador como um código transportável (Applet Java).

A Figura 17 mostra a arquitetura do sistema, composto por cinco camadas interdependentes: Módulos de Programas de Aplicação (NAPM – *NetTracker Application Modules*), Interface de Extensão de Aplicações (NEAPI – *NetTracker Extension Application Programming Interface*), Módulos de Extensão (NEM – *NetTracker Extensions Module*), Biblioteca de Objetos Java e Máquina Virtual Java.

Figura 17 Arquitetura do sistema NetTracker

A camada NAPM trata dos aspectos funcionais agregados ao sistema NetTracker, incorporando as aplicações responsáveis pela implementação das funções de gerenciamento e de comunicação com o administrador. Como exemplos de elementos componentes desta camada, podemos citar uma aplicação de Filtragem de *Traps*, juntamente com aplicações de gerenciamento de *trigger* e de busca e reconhecimento automático da topologia.

Na camada NEAPI, encontram-se disponíveis as interfaces de programação que fornecem acesso às funções implementadas pelos Módulos de Extensão. Todos os objetos responsáveis pelas funções básicas de apoio às tarefas de gerenciamento serão integrados à camada NEM. Tais funções desempenham o papel de alicerce para os programas de aplicações que se encontram na camada superior. Basicamente, esta é composta pelos sub-módulos: HTTPServer, Interface, Database, SNMP e Segurança.

Cada aplicação ou módulo de aplicação que pode ser integrado ao sistema é composto por um conjunto de definições de objetos que interagem entre si para proporcionar a funcionalidade esperada. Para tal, um destes objetos, dito objeto de ativação, deve ser definido obedecendo a duas regras essenciais:

- A definição de um objeto principal de aplicação deve necessariamente herdar, através da técnica de derivação, os atributos e métodos da superclasse *MVApplication*. Esta última, por sua vez, define os métodos abstratos de interfaceamento dos objetos do sistema, apresentados anteriormente, com os objetos de aplicação;
- Todo objeto de ativação, atue ele como uma *thread* ou não, deve implementar obrigatoriamente o método *run()*. Este método deve conter as instruções e operações que estabelecem, de fato, a aplicação no ambiente operacional do sistema, e será invocado logo após sua carga pelo mesmo. Em outras palavras, este método pode ser descrito como o corpo da aplicação.

Além da obrigatoriedade do cumprimento destas regras, e caso haja necessidade de se definir atributos de controle para a aplicação, cada um destes atributos devem ser definidos utilizando-se os objetos preestabelecidos para este propósito (*MVInteger*, *MVFloat*, *MVString*, *MVBoolean* e *MVByte*). É importante ainda ressaltar que cada atributo deverá registrar-se explicitamente, através do método *registerAttribute()*, no objeto de atributos (*MVApplicationAttributes*) da aplicação do qual fazem parte.

Seguindo estas regras na definição de uma aplicação, seu relacionamento com outros objetos no sistema se dá de forma transparente, mesmo quando as classes alvo do relacionamento ainda não estão carregadas e disponíveis. Neste caso, o próprio ambiente Java se incumbem de resolver tais dependências, carregando-as e ativando-as quando necessário.

B - Tecnologias Empregadas

Para viabilizar a execução deste projeto, diversos métodos e tecnologias foram utilizados. As próximas seções apresentam um breve resumo das principais ferramentas utilizadas, destacando linguagens de programação, recursos da linguagem Java¹¹.

Linguagens Utilizadas na Programação do Sistema

Java

Java é uma linguagem orientada a objetos projetada pela Sun Microsystems no ano de 1995. E como tal, fornece praticamente todas as características que definem uma linguagem orientada a objetos: encapsulamento, herança, polimorfismo, mensagens, etc. Outras características como persistência de objetos podem ser implementadas utilizando-se as novas interfaces de programação que estão sendo lançadas pela Sun.

Além dessas características específicas de linguagens orientadas a objetos, Java é uma linguagem interpretada, o que lhe fornece grande portabilidade. Os programas Java são compilados em *bytecodes* e são executados por uma máquina virtual Java (Java Virtual Machine).

Python

Python é uma linguagem de *scripts* projetada e desenvolvida inicialmente pelo

O fato de Python ser uma linguagem de *script* totalmente orientada a objetos deve ser destacado. Esta característica permite, por exemplo, criar a definição de uma classe dependendo do fluxo de execução de um programa. Além disto, Python possui uma vasta gama de módulos que

¹¹ As referências para os tópicos deste apêndice são documentos disponíveis nos sites <http://java.sun.com> e <http://www.python.org>.

implementam acesso a serviços do sistema, tais como: correio eletrônico, acesso a base de dados e gerenciamento de redes.

Uma implementação de Python totalmente feita em Java está disponível sob a denominação JPython.

Recursos da Linguagem Java

Introspecção ou Reflexão

A introspecção ou reflexão é a característica da linguagem Java que permite a obtenção de informações a respeito de objetos e classes de objetos. Através de reflexão é que se torna possível implementar geradores de aplicação avançados baseados no padrão JavaBeans.

RMI - Remote Method Invocation

RMI é um modelo de distribuição de objetos que aproveita as características especiais do modelo de objetos da linguagem Java. Os objetivos primordiais da RMI são:

1. suportar invocação transparente de métodos de objetos em diferentes máquinas virtuais;
2. suportar chamadas dos servidores para os applets;

integrar o modelo de objetos distribuídos na linguagem Java de modo natural, enquanto mantém a semântica dos objetos da linguagem Java;

3. tornar as diferenças entre o modelo de objetos distribuídos e o modelo de objetos local do Java aparentes;
4. tornar a construção de aplicações distribuídas confiáveis tão simples quanto possível;
5. preservar a segurança fornecida pelo ambiente de execução do Java.

Sob todas essas metas está um requisito geral de que o modelo RMI seja ao mesmo tempo simples (fácil de usar) e natural (molde-se bem a linguagem).

Ainda, o sistema RMI deveria permitir extensões como coleta de lixo de objetos remotos, replicação de servidores e ativação de objetos persistentes para servir a uma invocação. Estas extensões deveriam ser transparentes para o cliente e adicionar dificuldades mínimas para os servidores que as utilizarem. Para suportar tais extensões, o sistema deveria também suportar:

1. vários mecanismos de invocação, por exemplo: invocação simples a um único objeto ou invocação a um objeto replicado em múltiplas localidades. O sistema deve também ser extensível por outros paradigmas de invocação;
2. várias semânticas de referência para objetos remotos, por exemplo: referências vivas (não-persistentes), referências persistentes e ativação retardada;
3. “coleta de lixo” distribuída dos objetos ativos;
4. capacidade de suportar múltiplos transportes.

Em termos de arquitetura, o sistema RMI consiste de três camadas:

1. camada “*stub/skeleton*” - *stubs (proxies)* do cliente e *skeletons* no servidor;
2. camada de referência remota - comportamento de referência remota (e.g., invocação de um objeto único ou replicado);
3. camada de transporte - inicialização e controle da conexão e monitoração de objetos remotos.

A camada de aplicação fica sobre o sistema RMI. O relacionamento entre as camadas é demonstrado na Figura 18.

Figura 18 - Arquitetura de uma aplicação RMI

JCA (Java Cryptography Architecture) e JCE (Java Cryptography Extension)

Muitas aplicações do mundo real necessitam utilizar recursos de criptografia como: *message digest*, assinatura digital, certificados e criptografia de chave pública. Para atender a demanda destas aplicações, a Javasoft definiu uma arquitetura de suporte a criptografia para a linguagem Java, a JCA. Através de técnicas avançadas de projeto e implementação, esta arquitetura permite que o usuário utilize facilmente implementações de algoritmos provindas de vários fornecedores. A vantagem de tal abordagem é permitir que o usuário tenha o nível de qualidade de serviço necessário a sua aplicação, por exemplo, se a implementação do algoritmo SHA da Sun não atende aos requisitos de desempenho de uma aplicação, o programador pode facilmente embutir a implementação de uma empresa qualquer, sem a necessidade de alteração de muito código