Avaliação do Uso de Agentes Móveis em Segurança Computacional

Mauro Cesar Bernardes

Orientador: Prof. Dr. Edson dos Santos Moreira

Dissertação entregue ao Instituto de Ciências Matemáticas e de Computação - ICMC - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Área: Ciências de Computação e Matemática Computacional.

São Carlos
Novembro/1999

Aos meus pais, irmão, namorada e amigos, minha maior riqueza.

Aos 20 milhões de brasileiros analfabetos que, infelizmente, não tiveram a mesma sorte que eu.

Agradecimentos

Deus, obrigado por me conceder mais essa conquista e ainda, por colocar pessoas tão especiais em minha vida:

Prof. Dr. Edson dos Santos Moreira: Meu Orientador! Obrigado pelo voto de confiança, imenso apoio, motivação, amizade e conselhos indispensáveis a este trabalho.

Elaine: Você foi com certeza a pessoa que mais torceu para que esse dia chegasse. Devo a você muitos incentivos, votos de confiança, carinho, amor, companheirismo e tudo que me fez muito feliz ao longo destes anos. Sem você, com certeza não teria chegado até aqui. Obrigado por me esperar!

Edmilson: Além de amigo, companheiro e guia (em muitos sentidos que a palavra possa expressar) você se demonstrou um grande irmão, ouvindo meus desabafos, motivando-me, agüentando as pontas ao volante enquanto por muitas vezes (a maioria) eu dormia, encorajando-me, etc. Seu espírito evoluído contribuiu muito para meu crescimento.

Prof. Alexandre e Profa. Marly: Grandes amigos: Vocês foram os maiores incentivadores desta jornada. Obrigado por entenderem minhas ausências.

Leonardo: Outro grande irmão e companheiro de república. Nossas constantes conversas que por muitas vezes (e necessariamente) me faziam "desligar" das atividades, marcaram uma fase muito boa de minha vida.

Tatiana: Amiga e companheira. Que nossa linha não se arrebente!

Cláudio Hirose (*"Irmãozinho"):* sempre pronto a ajudar, agradeço-lhe por toda atenção prestada.

Stênio: Meu ex-aluno, amigo e companheiro de grupo de Segurança Computacional do ICMC. Devo a você toda a garra desse *sprint* final.

Milagres, Brandão e Daniel Ambrósio: Colegas do Grupo de Segurança Computacional. Obrigado pelos conselhos úteis em nossas reuniões, darei o máximo de mim para recompensá-los no desenvolvimento de vossos trabalhos.

Amigos: O que seria de mim sem o apoio de vocês. Desculpem-me pelo abandono ao longo dos dois últimos anos.

Beth, Marília, Laura e Paulinho: funcinários exemplares. Obrigado pela simpatia, atenção, presteza e educação. Nosso país precisa de mais gente como vocês.

Alunos da Universidade de Alfenas (em especial aos meus orientandos de iniciação científica Patrícia, Heber, Pablo, Leonardo, Fabrício e José Maria): vocês tiveram paciência suficiente para compreender minhas ausências e continuarem os trabalhos. Obrigado!

Colegas do ICMC: Obrigado a todos vocês por me fazerem sentir em casa.

UNIFENAS: Meu lar! Obrigado pelo carinho ao longo destes 9 anos.

A todos que torceram para este dia: pois com certeza não chegaria até aqui sem a força e carinho de vocês.

A todos que torceram para que este dia não chegasse: pois, além de me servirem de grande inspiração e desejos de vencer, experimentam hoje o sabor de mais uma derrota.

Sumário

Lista de Figuras

Lista de Tabelas

RESUMO

Em decorrência do aumento do número de ataques de origem interna, a utilização de mecanismos de proteção, como o *firewall*, deve ser ampliada. Visto que este tipo de ataque, ocasionado pelos usuários internos ao sistema, não permite a localização imediata, torna-se necessário o uso integrado de diversas tecnologias para aumentar a capacidade de defesa de um sistema. Desta forma, a introdução de agentes móveis em apoio a segurança computacional apresenta-se como uma solução natural, uma vez que permitirá a distribuição de tarefas de monitoramento do sistema e automatização do processo de tomada de decisão, no caso de ausência do administrador humano. Este trabalho apresenta uma avaliação do uso do mecanismo de agentes móveis para acrescentar características de mobilidade ao processo de monitoração de intrusão em sistemas computacionais. Uma abordagem modular é proposta, onde agentes pequenos e independentes monitoram o sistema. Esta abordagem apresenta significantes vantagens em termos de overhead, escalabilidade e flexibilidade

ABSTRACT

The use of protection mechanisms must be improved due the increase of attacks from internal sources. As this kind of attack, made by internal users do not allow its immediate localization, it is necessary the integrated use of several technologies to enhance the defense capabilities of a system. Therefore, the introduction of mobile agents to provide security appears to be a natural solution. It will allow the distribution of the system monitoring tasks and automate the decision making process, in the absence of a human administrator. This project presents an evaluation of the use of mobile agents to add mobile capabilities to the process of intrusion detection in computer systems. A finer-grained approach is proposed, where small and independent agents monitor the system. This approach has significant advantages in terms of overhead, scalability and flexibility.

1 Introdução

Às portas de um novo milênio, é reconhecido que a informação é a fonte vital na busca pelo conhecimento e, por conseguinte, pelo poder. Paralelamente, a obtenção, manutenção e disseminação da informação tornam-se uma das grandes preocupações da sociedade em geral, ocasionando uma crescente expansão das formas de armazenamento e de distribuição através de redes de computadores.

Em contrapartida, tentativas de ataque e invasões consumadas tornam-se freqüentes, envolvendo um número crescente de computadores. Desta forma, segurança torna-se uma das palavras de ordem para a maioria absoluta das empresas em todo o mundo. Com a utilização cada vez maior da tecnologia Internet em ambiente corporativo (as *Intranets*) e sua abertura para o mundo externo (*Extranets*) para atividades essenciais, a preocupação e o risco de invasão dos sistemas empresariais cresceram muito nos últimos anos. E não haveria de se esperar algo diferente: os crimes digitais são, na maioria das vezes, muito difíceis de serem descobertos e até mesmo rastreados. Prova disso, é que muitas empresas sofrem invasões em seus sistemas e só se dão conta do fato muito tempo depois, isso quando descobrem.

Mas o panorama esta mudando a rapidamente. A cada dia surgem soluções mais robustas de proteção dos dados. *Firewalls*, criptografia de vários bits, certificados digitais, VPNs (*Virtual Private Networks*), *smart cards* e até mesmo biometria (reconhecimento de alguma parte do corpo, como íris dos olhos ou a palma da mão) já fazem parte do arsenal utilizado no combate à violação de sistemas e credibilidade das transações *on-line*.

A tecnologia de segurança de rede mais utilizada hoje em dia é o *firewall*. Este sistema previne a entrada não autorizada utilizando-se de mecanismos de controle de acesso externo. Porém não existe nenhum sistema que possa ser considerado a panacéia em matéria de proteção e ainda, que forneça um elevado grau de segurança enquanto permite uma certa flexibilidade e liberdade no uso dos recursos computacionais.

Existem fatores que tornam muito difícil impedir que atacantes eventualmente tenham acesso a um sistema. A maioria dos computadores possui algum tipo de “furo de segurança” que permite a atacantes externos (ou ainda usuários internos legítimos) terem acesso a informações confidenciais. Mesmo um sistema supostamente seguro pode ser vulnerável a usuários internos abusando de seus privilégios ou ser comprometido por práticas impróprias. Em vista disto, uma vez que um ataque pode ser considerado inevitável, existe uma óbvia necessidade por mecanismos que possam detectar atacantes tentando penetrar no sistema ou usuários legítimos fazendo mal uso de seus privilégios.

Com o crescente aumento no número de ataques internos, a utilização de mecanismos como o *firewall* deve ser ampliada. Visto que este tipo de ataque, ocasionado pelos próprios usuários do sistema, não permite a localização imediata, torna-se necessário o uso integrado de diversas tecnologias para aumentar a capacidade de defesa de um *site*. Entre estas tecnologias, torna-se interessante a presença de mecanismos que acrescentem características de mobilidade no processo de monitoração do sistema. Desta forma, a introdução de agentes móveis em apoio à segurança computacional apresenta-se como uma solução natural, uma vez que permitirá a distribuição de tarefas de monitoramento do sistema e a agilização no processo de tomada de decisão no caso de ausência do administrador humano.

Desde 1990, diversos trabalhos nas áreas de gerenciamento e de segurança de redes foram realizados pelo Grupo de Segurança Computacional do *Laboratório Intermídia do ICMC-USP*. A continuidade dos projetos e a pesquisa de métodos alternativos de gerenciamento e de segurança de redes têm sido uma preocupação constante. Nesse contexto, a presente proposta de trabalho representa mais uma possibilidade de expansão do conhecimento teórico e prático conseguido 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 [LIEIRA, 1995] [MORAES, 1995] e o desenvolvimento de um agente SNMP [CICILIANI, 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, consiste na integração de serviços de detecção de intrusos ao NetTracker. Um sistema de detecção de intrusos baseado em redes neurais [CANSIAN, 1997a] [BONIFÁCIO et al., 1998] já foi desenvolvido e adaptado para que seja possível utilizá-lo através do ambiente NetTracker. Também nesta fase, foi especificado e desenvolvido um protótipo um ambiente de gerenciamento de segurança apoiado por agentes móveis [REAMI,1998]

Atualmente, o grupo de segurança computacional encontra-se em um estágio avançado para o desenvolvimento de um *Sistema de Detecção de Intrusão* baseado em Agentes Móveis. Este trabalho, apresenta uma modelagem para as especificações apresentadas na tese de mestrado de Elderclei Regis Reami [REAMI, 1998], modela alguns cenários de utilização do Ambiente e ainda, implementa um cenário de verificação de anomalias.

1.2 Organização da Monografia

O texto desta monografia está organizado da seguinte maneira:

- O Capítulo 2 apresenta os principais aspectos de segurança relacionados a redes de computadores, fornece um breve resumo das técnicas de detecção de intrusão e aborda alguns sistemas de detecção de intrusão apoiados por diversas técnicas;

- O Capítulo 3 faz uma revisão sobre o conceito de agentes, incluindo uma tipologia aceitável para a estrutura de agentes; Apresenta as principais características de Agentes Móveis comparando-os com os tradicionais mecanismos cliente/servidor e ainda, faz uma avaliação de sua utilização em um ambiente de segurança computacional;

- O Capítulo 5 apresenta a proposta para uma arquitetura baseada em camadas para um SDI não-monolítico apoiado em agentes móveis e a modelagem para alguns cenários de utilização do sistema.

- O Capítulo 6 discute as principais tecnologias para o desenvolvimento de um ambiente computacional apoiado por agentes móveis, explorando recursos das linguagens de programação para desenvolvimento de sistemas apoiados por agentes móveis, fazendo um *overview* dos principais ambientes de desenvolvimento e hospedagem e terminando com a apresentação da padronização de interoperabilidade aceita recentemente pela OMG: o MASIF.

- O Capítulo 7 detalha a implementação de um cenário para o sistema utilizando o ASDK IBM.

- O Capítulo 8 apresenta as conclusões para o trabalho e a proposta para trabalhos futuros;

2 Segurança Computacional

2.1 Considerações Iniciais

O termo segurança é usado com o significado de minimizar a vulnerabilidade de bens (qualquer coisa de valor) e recursos. Vulnerabilidade é qualquer fraqueza que pode ser explorada para se violar um sistema ou as informações que ele contém [ISO, 1989].

A segurança está relacionada à necessidade de proteção contra o acesso ou manipulação, intencional ou não, de informações confidenciais por elementos não autorizados, e a utilização não autorizada do computador ou de seus dispositivos periféricos. A necessidade de proteção deve ser definida em termos das possíveis ameaças e riscos e dos objetivos de uma organização, formalizada nos termos de uma política de segurança [SOARES, 1995]. Zorkle e Levitt ainda acrescentam que a segurança depende de mais do que a integridade do software e mecanismos de proteção do sistema operacional em uso; ela também é dependente da própria configuração e uso do software [ZERKLE & LEVITT, 1996].

2.2 Intrusos e Aspectos de Segurança

Segurança de redes de computadores é uma área de crescente interesse e preocupação, atingindo desde administradores preocupados com a segurança e o bom funcionamento de seus *sites* até *hackers* e vândalos buscando novos métodos e técnicas de ataque. O termo *hacker* deriva da década de 70, quando designava pessoas que possuíam um profundo conhecimento sobre computadores, sistemas operacionais e softwares, não tendo nenhuma ligação com os atuais significados no que se refere a atacantes e intrusos. Geralmente os termos *hacker* e *cracker* são usados indiscriminadamente, mas algumas diferenciações são encontradas na literatura.

Hacker é o indivíduo com um profundo conhecimento, mas geralmente sem intenções destrutivas. Seu propósito é unicamente provar que consegue invadir um determinado sistema e quanto mais protegido for este sistema, maior será seu empenho. De forma oposta, o *cracker* é aquele cujo único objetivo é destruir, danificar e causar perdas. Um estereótipo típico criado é o

de um adolescente em sua casa que, a partir de um computador e um modem, profere ataques aos computadores de grandes organizações.

A segurança de uma rede de computadores pode ser comparada à segurança de uma casa. Não importa que grau de segurança exista, não importa que sistemas ou trancas sejam usados. Quando alguém decide com suficiente empenho, invadir, provavelmente terá êxito. De modo análogo, todas as medidas no sentido de se aumentar a segurança de uma rede tem como objetivo torná-la tão segura quanto possível, já que nenhum sistema conhecido garante o estado-da-arte em termos de proteção. Geralmente, um atacante irá analisar a relação custo/benefício, ou seja, o quão custoso e complicado será invadir um determinado sistema ponderado aos lucros que ele alcançará com tal invasão. Uma vez que esta proporção se torne inviável, pode-se dizer que foi atingido um bom grau de segurança [BONIFÁCIO, 1998].

Ao mesmo tempo em que a Internet é o meio pelo qual a maioria das intrusões e ataques ocorrem, é também através dela que são largamente disponibilizados e veiculados documentos explicando e demonstrando técnicas de *hacking*, furos de segurança e casos de monitoração de intrusões em andamento. É possível encontrar com facilidade documentos do tipo “receita de bolo”, que ensinam passo a passo técnicas de intrusão. Ainda que sejam técnicas simples, podem ser altamente destrutivas, tendo-se em vista que boa parte das redes conectadas à Internet não possuem um completo domínio das questões relacionadas à segurança.

Existem atualmente diversos sites dedicados exclusivamente a este assunto, contendo documentos que abordam desde técnicas básicas de *hacking* até conceitos avançados para se aumentar a segurança de uma rede. Exemplos de site são: *http://www.rootshell.com* (visitado em 10/12/1998) e *http://www.underground.org* (visitado em 29/01/1999). Semanalmente são divulgados relatórios com novos furos de segurança nos mais variados sistemas operacionais e softwares. Como existe uma certa demora no lançamento de *patches* de segurança e uma grande dificuldade dos administradores se manterem constantemente atualizados, tem-se um cenário em que uma rede pode-se alcançar um estado altamente vulnerável muito rapidamente.

Esta conjuntura de computadores, redes e comunicações inseguras deve-se, em parte, ao modo como a Internet foi projetada. O principal foco do projeto da Internet, e mais basicamente do protocolo TCP/IP, estava muito distante dos atuais usos da Internet. Seu projeto previa inicialmente o uso por instituições militares e de pesquisa. O crescimento e a popularização da

Internet, o surgimento de aplicações de comércio eletrônico, a interligação das redes das diversas filiais de uma empresa e muitas outras características e serviços oferecidos pela Internet de hoje, não eram sequer supostas pelos seus projetistas e técnicos. O crescimento da Internet levou a uma mudança no foco e no perfil dos usuários e das aplicações da rede. Em vista disso, os protocolos, serviços, sistemas operacionais como o UNIX e os softwares que são utilizados na Internet não foram projetados e especificados com as devidas preocupações com relação à segurança. No UNIX, as senhas circulam totalmente abertas pela rede, o protocolo TCP/IP não prevê nenhum esquema de criptografia dos dados ou autenticação das máquinas e usuários envolvidos em uma conexão. Os atuais sistemas, como o Windows NT, foram criados em cenários com preocupações específicas sobre segurança. Porém ainda não se mostraram soluções totalmente confiáveis devido a fatores como pouco tempo para desenvolvimento e testes, o que acarretou em falhas e furos de segurança.

Segurança de redes é um assunto muito vasto e é interessante dividi-lo em duas sub-áreas: Segurança do *site* e segurança nas comunicações.

2.2.1 Segurança do Site

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.

Reami [REAMI, 1998] apresenta quatro princípios básicos para a segurança de um *site:*

- **Confidencialidade:** Apenas quem tem os direitos de acessar um determinado recurso ou informação poderá efetivamente acessá-los.

- **Integridade:** Garante que os dados armazenados não serão alterados, tanto como conseqüência de atos provenientes de uma 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.

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

- **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).

Um *backdoor* é um tipo de ataque muito comum em que um software inserido, ou modificado em uma rede pode, a partir de uma combinação especial de caracteres ou a um evento de tempo, ter um comportamento diferente do esperado. Um dos rastros deixados por uma invasão geralmente são modificações em programas como o *deamon* de *telnet*, em que ele é programado para quando alguém entrar com o *login hacker*, por exemplo, não seja pedida a senha e o acesso seja liberado. Já o *Trojan Horse*, outro ataque clássico, consiste em se trocar o processo de *login* por outro programa, de comportamento idêntico. Este programa pede o *username* e a senha do usuário, salva-o, exibe uma mensagem de erro de senha e chama o verdadeiro processo de *login*, a partir do qual tudo transcorre normalmente, a não ser pelo fato de que a senha do usuário foi capturada. Para o usuário, tudo ocorre normalmente, ele apenas pensa que digitou errado sua senha.

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, os administradores não conseguem dar a devida importância à segurança de rede, quer seja por falta de informação, por falta de tempo ou ainda, por desinteresse. As falhas mais comuns oriundas de administradores são: não instalação de sistemas de proteção

e auditoria; não aplicação de *patches* de problemas conhecidos; negligenciar procedimentos básicos de segurança; não orientar os usuários e não se manterem 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 a 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.

A manutenção de senhas é um fator que está fortemente relacionada à quebra de segurança. As senhas em sistemas UNIX são guardadas no arquivo */etc/passwd* cifradas. Porém o algoritmo usado, chamado de *one-way*, não permite que a partir da senha cifrada, se obtenha de volta a senha original. Desta forma, o sistema pega a senha do usuário, cifra-a e compara-a com a que tem guardada. Um intruso, mesmo com acesso ao arquivo de senhas, não tem acesso às senhas originais e, portanto, não poderá entrar no sistema. Porém, existe uma técnica chamada de quebra de senha por força bruta na qual o intruso, através de programas chamados *de cracker de senhas*, escolhe palavras em um dicionário, faz combinações com números, maiúsculas e minúsculas, cifra uma por uma e compara com as senhas cifradas no arquivo de senhas. Quando ele conseguir uma igual, ele terá descoberto uma senha válida para o sistema. Senhas fracas são senhas que podem ser facilmente quebradas, como nomes de pessoas, palavras comuns ou o próprio *username.* Se todos os usuários possuírem senhas fortes, a chance de que um ataque baseado na captura do arquivo de senhas tenha sucesso seriam menores.Outra grande ameaça são os ataques externos. Ataques externos geralmente se dão de quatro formas diferentes (apresentado na figura 2-1), podendo partir de um único atacante ou um grande grupo. Um ataque pode ocorrer a partir de uma única máquina atacando outra (a), uma máquina sendo atacada por diversas outras (b), várias máquinas sendo atacadas a partir de uma única (c), ou ainda, ser um ataque indireto onde o atacante ataca outra máquina para então realizar o ataque ao seu alvo principal (d).

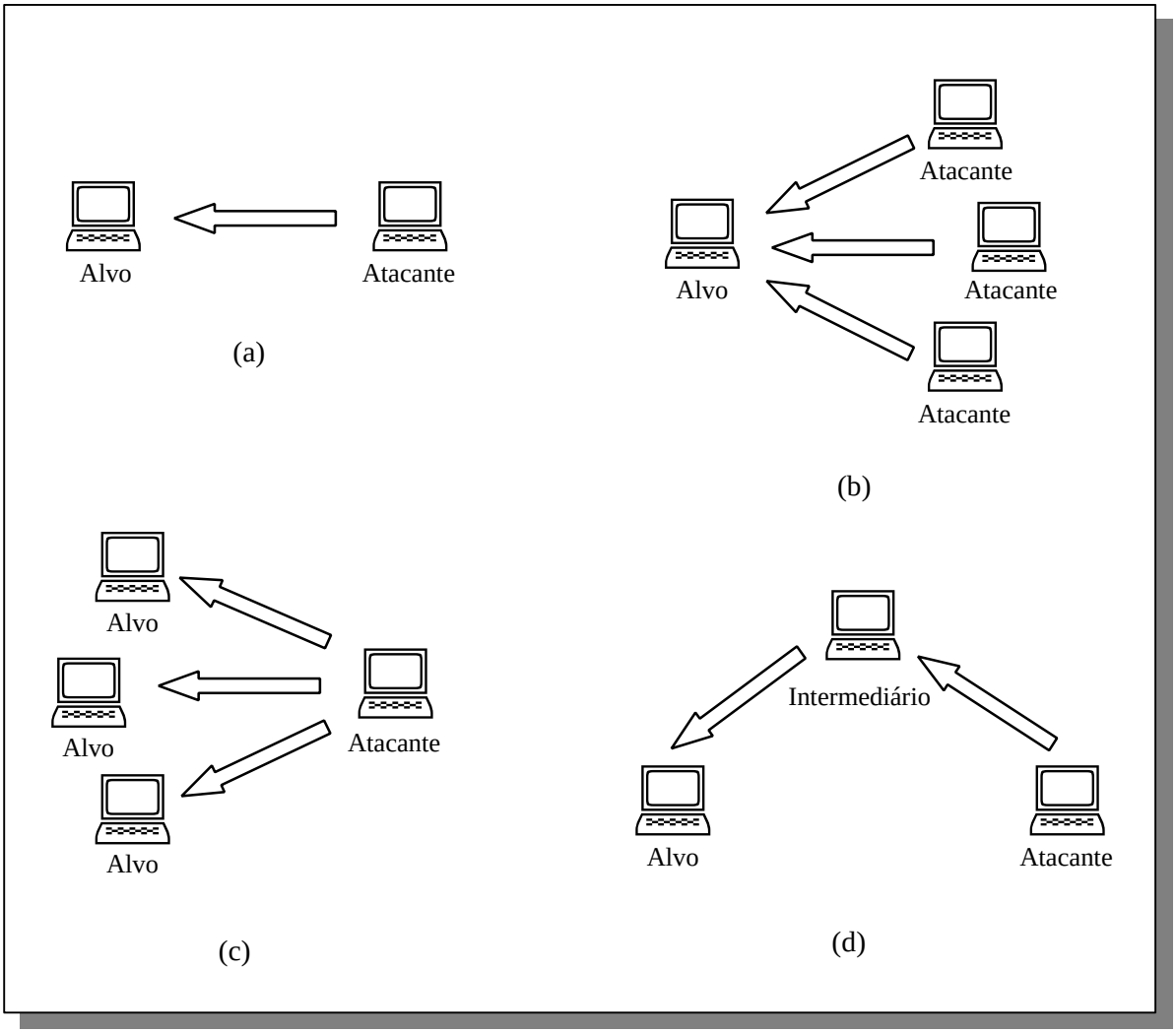

Figura 2-1- Tipos de Ataques Externos

2.2.2 Segurança nas Comunicações

Conceitos de segurança relacionados às comunicações tratam das informações que estão trafegando na rede. Como o protocolo TCP/IP é inerentemente inseguro [BELOVINS, 1989], diversos recursos têm de ser utilizados em conjunto para se aumentar a segurança que o protocolo oferece. As informações que trafegam pela rede através de TCP/IP são abertas, ou seja, qualquer pessoa com acesso ao meio físico pode ter acesso aos dados que estiverem trafegando, bastando para isso, ter uma interface de rede em modo promíscuo, ou seja, todos os pacotes serão capturados, não importando se são destinados a ela ou não. Esta técnica é chamada de *sniffing*.

Um intruso pode ser classificado em passivo ou ativo. Um intruso passivo simplesmente monitora a rede em busca de informações como senhas, números de cartões de crédito e informações confidenciais; enquanto que o intruso ativo atua modificando o conteúdo dos pacotes.

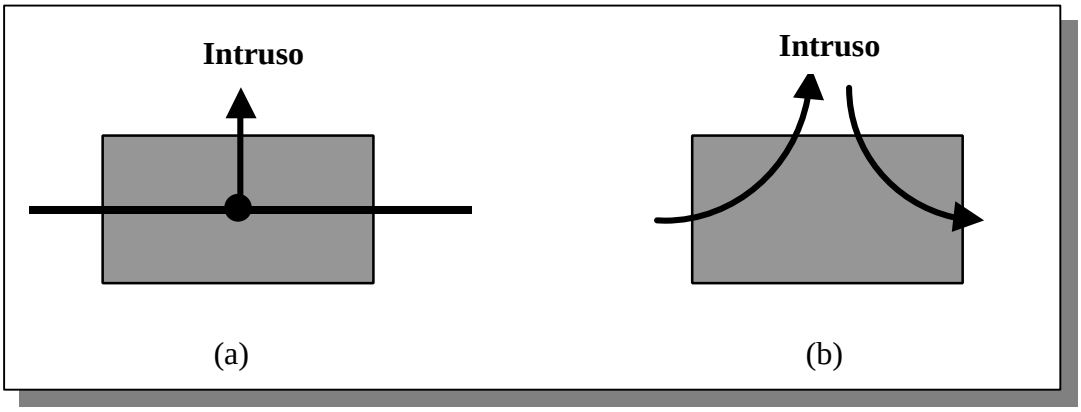

Figura 2-2 - Intruso passivo (a) e intruso ativo (b)

Desta forma, é preciso alguns requisitos para se garantir um ambiente em que se tenha segurança nas comunicações em redes de computadores:

- **Confidencialidade:** Apenas as partes envolvidas podem ter acesso ao conteúdo dos dados que estão trafegando na rede. Qualquer ação de monitoria da rede não deve ser capaz de ter acesso aos dados.

- **Integridade:** Deve-se garantir que a informação transmitida em um ponto é a mesma recebida em outro e que não houve qualquer adulteração dos dados por partes de terceiros ou falhas.

- **Autenticidade:** As partes envolvidas em uma comunicação devem ter meios de confirmarem mutuamente suas identidades, certificando-se de com quem estão se comunicando.

De forma semelhante à segurança do *site*, não existe nenhum protocolo ou solução completamente segura. Novos protocolos têm sido propostos e utilizados no intuito de se atingir um alto grau de segurança nas comunicações, protocolos como HTTPS e SSL são voltados a aplicações *WEB*, enquanto que o protocolo SET (*Secure Eletronic Transaction*) [SET, 1999]

recentemente desenvolvido por grandes empresas como IBM e outras em parceria com a VISA, visam atacar o problema do comércio eletrônico, onde existem três partes envolvidas na negociação: o cliente, o vendedor e o banco onde será efetuado o pagamento.

2.2.3 Criptografia

A principal técnica de segurança utilizada para garantir segurança nas comunicações é a criptografia [TANEMBAUM, 1997]. A criptografia surgiu da necessidade de se enviar informações sigilosas através de meios de comunicação não confiáveis, ou seja, em meios onde não é possível garantir que um intruso não irá interceptar o fluxo de dados para leitura ou para modificá-lo. Criptografia consiste em técnicas que permitem transformar um texto legível em outro, segundo um algoritmo, de forma que a obtenção do texto original a partir do cifrado seja possível apenas usando-se o mesmo algoritmo. Criptografia teve seu grande desenvolvimento durante a segunda guerra mundial e a guerra fria para garantir que o inimigo não tivesse acesso às comunicações. Entretanto, encontramos relatos de sua utilização desde muito tempo atrás.

A técnica mais simples é a transposição de letras, por exemplo, trocando-se cada letra por sua subsequente no alfabeto: USP → VTQ. Claro que a criptografia usada nos modernos sistemas utilizam algoritmos muito mais complexos baseados em chaves. Após o texto ser cifrado (*plaintext*) é gerado um *ciphertext* usando-se como parâmetros uma chave como na Figura 2-3. Existem dois tipos principais: criptografia com chave simétrica e criptografia com chaves públicas. Como a recuperação da informação original está vinculada ao conhecimento da chave, um processo para se conseguir quebrar o texto cifrado é o mesmo usado com senhas: busca exaustiva. Neste caso, são geradas todas as possíveis combinações de chaves até que se ache a correta. O grande problema desta solução é que o número de chaves possíveis cresce exponencialmente com o tamanho da chave e as chaves usadas em aplicações militares ou de alta segurança podem levar alguns milhares de anos para serem quebradas com os recursos computacionais disponíveis atualmente.

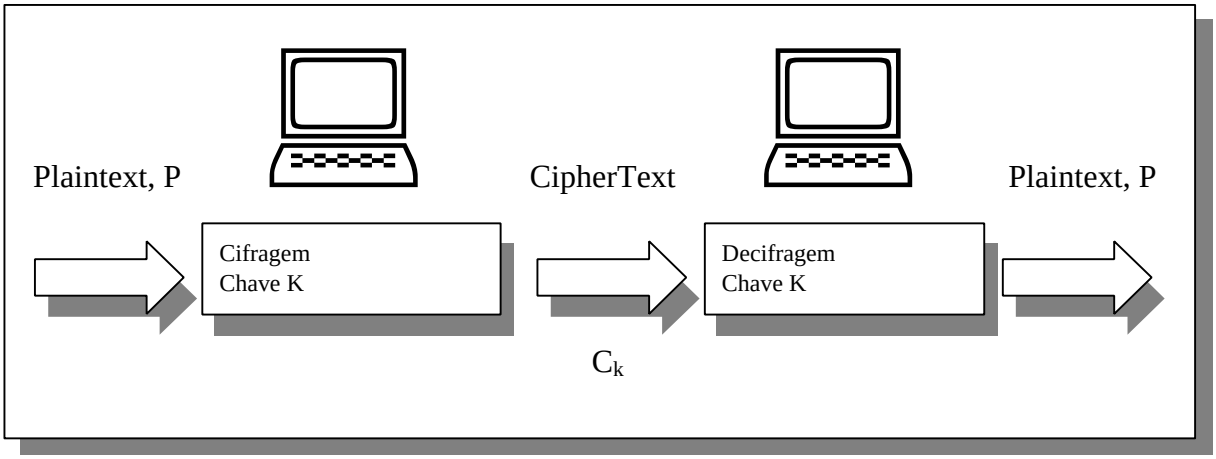

Figura 2-3 - Criptografia Baseada em Chaves

Na criptografia com chaves simétricas, é usada a mesma chave para cifrar e decifrar um texto. Neste caso, ambas as partes envolvidas tem de concordar com uma chave antes de iniciar a comunicação, o que pode ser problemático, pois a menos que se conheçam, a escolha da chave deve ser feita usando-se a própria rede insegura. Exemplos de algoritmos de chave simétrica são DES (*Data Encryption Standard*), largamente utilizado no passado pelo governo americano, TrippleDES e IDEA (*International Data Encryption Algorihm)* [TANEMBAUM, 1997].

A criptografia de chave pública utiliza duas chaves diferentes e complementares. O que é cifrado com uma chave só pode ser decifrado com a outra e vice-versa. O usuário deixa uma chave de conhecimento público (chave pública) e mantém a outra em segredo (chave privada). Este modelo evita as duas partes terem de concordar com uma única chave, uma vez que cada uma conhece a chave pública da outra, podendo com ela, cifrar os textos que só poderão ser lidos pela chave privada correspondente. Algoritmos de chave pública são o RSA (*Rivest, Shamir e Adleman,* as iniciais dos inventores) e El Gamal, por exemplo.

Com o uso de criptografia, seja por chave simétrica ou pública, pode-se garantir a *confidencialidade* dos dados que serão enviados em uma conexão. Apesar da criptografia de chave pública ser mais segura que a de chave simétrica, ela é muito lenta para grandes volumes de dados. Atualmente, soluções como o protocolo SSL (*Security Socket Layer*) da *Netscape Comunications* [SSL, 1996] utiliza as duas técnicas em conjunto para se assegurar a velocidade. O primeiro passo é usar criptografia de chave pública para que as duas partes possam concordar

em uma chave que posteriormente será usada na criptografia de chave simétrica durante todo o restante da comunicação.

O uso de criptografia garante apenas a confidencialidade dos dados, uma vez que um simples ataque do tipo *man-in-the-middle*, onde uma terceira pessoa está posicionada entre as duas partes que querem se comunicar faz com que cada uma delas acredite estar falando com a outra, sendo que estão ambas se comunicando com esta terceira pessoa. Ou seja, não há meios de se garantir que uma determinada chave pública pertença realmente a alguém. Para isso, existem os certificados digitais. Através desta assinatura, um usuário pode confirmar se uma determinada chave pública pertence realmente ao seu dono. Certificados digitais estão sendo largamente usados nos *browsers Web* (principalmente no Netscape e no Internet Explorer) e se baseiam no conceito de entidades certificadoras que emitem os certificados baseados em suas chaves pública e privada. Como os *browsers* já tem a chave pública de um certificador, e no qual eles confiam, eles podem verificar a autenticidade do portador de um certificado gerado pelo certificador, garantindo-se a autenticidade. Em um certificado digital do tipo X.509 utilizado pelo SSL, vão informações como o nome do dono, data de criação, data de validade, versão e a chave pública do dono cifrada através da chave privada do certificador. Desta forma, garante-se que apenas a chave pública do certificador possa decifrar a chave pública e esta não precisa ser transferida pela rede sem garantias.

2.3 Ameaças e Ataques

Uma ameaça consiste em uma possível violação da segurança de um sistema. Algumas das principais ameaças às redes de computadores são:

- Destruição de informação ou de outros recursos;
- Modificação ou deturpação da informação;
- Roubo, remoção ou perda de informação ou de outros recursos;
- Revelação de informação;
- Interrupção de serviços.

As ameaças podem ser classificadas como acidentais ou intencionais, podendo ambas serem ativas ou passivas. Ameaças intencionais são as que não estão associadas à intenção premeditada (descuidos operacionais, *bugs* de software ou hardware). A concretização das ameaças intencionais varia desde a observação de dados com ferramentas simples de monitoramento de redes, a ataques sofisticados baseados no conhecimento do funcionamento do sistema. A realização de uma ameaça intencional configura um ataque.

Ameaças passivas são as que, quando realizadas, não resultam em qualquer modificação nas informações contidas em um sistema, em sua operação ou em seu estado. Uma realização de uma ameaça ativa a um sistema envolve a alteração da informação contida no sistema, ou modificações em seu estado ou operação.

Alguns dos principais ataques que podem ocorrer em um ambiente de processamento e comunicações de dados são os seguintes:

- **Personificação**: uma entidade faz-se passar por outra a fim de obter privilégios extras;
- **Replay**: uma mensagem, ou parte dela, é interceptada e posteriormente transmitida para produzir um efeito não autorizado;
- **Modificação**: o conteúdo de uma mensagem é alterado, implicando em efeitos não autorizados sem que o sistema consiga detectar a alteração;
- **Recusa** ou **Impedimento do Serviço**: ocorre quando uma entidade não executa sua função apropriadamente ou atua de forma a impedir que outras entidades executem suas funções;
- **Ataques internos**: ocorrem quando usuários legítimos comportam-se de modo não autorizado ou não esperado;
- **Ataques externos**: ocorrem quando usuários externos ou pessoas não autorizadas conseguem uma conexão externa e realizam ações inesperadas.
- **Armadilhas**: ocorre quando uma entidade do sistema é modificada para produzir efeitos não autorizados em resposta a um comando, a um evento, ou seqüência de eventos predeterminados.

- **Cavalos de Tróia**: nesse ataque, uma entidade executa funções não autorizadas, em adição às que está autorizada a executar.

2.4 Firewalls

Um mecanismo muito utilizado na prática para aumentar a segurança das redes de computadores, protegendo-as de ataques externos, é o *firewall.* Um *firewall* fundamenta-se no fato de que normalmente a segurança é inversamente proporcional a complexidade. Assim, proteger máquinas de uso geral onde são executados diferentes aplicações, de variados portes, é uma tarefa complicada, pois é muito improvável que nenhuma das várias aplicações apresente falhas que possam ser exploradas para violar a segurança do sistema. Desta forma, fica muito mais fácil garantir a segurança isolando as máquinas de uso geral de acessos externos, usando uma barreira de proteção que impeça a exploração das possíveis falhas.

Na configuração de um *firewall*, as principais decisões relacionadas à segurança são freqüentemente ditadas pela política de segurança da organização ou corporação. Especificamente, as decisões devem ser tomadas pensando-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. O princípio da simplicidade tem como conseqüência a seguinte consideração: para diminuir os riscos, a configuração dos *firewalls* deve ser minimizada, excluindo tudo que não seja estritamente necessário.

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 e os serviços devem ser permitidos caso a caso após um cuidadoso estudo de necessidade e risco. Isso causa impacto imediato nos usuários, que podem ver o *firewall* como um incomodo. Essa 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.

Um *firewall*, em geral, consiste nos componentes mostrados na Figura 2-4. Os filtros (*screens*) bloqueiam a transmissão de certas classes de tráfego. O componente *gateway* é uma máquina, ou um conjunto de máquinas conectadas por um segmento de rede, que fornecem serviços de retransmissão. O filtro colocado na saída (entre a rede externa e o *gateway*) é usado para proteger o *gateway* de ataques externos, enquanto o filtro interno protege a rede interna das conseqüências de um ataque que tenha conseguido comprometer o funcionamento do *gateway*. Assim, os dois filtros atuando isoladamente, ou em conjunto, protegem a rede interna de ataques externos. Um *gateway* do *firewall* que pode ser acessado a partir da rede externa é chamado de *bastion host.* Cabe reafirmar que, fisicamente, os filtros e o *gateway* podem ser implementados em uma única máquina, ou em um conjunto de máquinas ligadas por um segmento de rede [SOARES, 1995].

Outras formas de configuração são apresentadas na literatura pesquisada. Cada uma apresenta seu ponto forte e também sua fragilidade. Informações mais detalhadas podem ser encontradas em [CHAPMAN, 1992][WACK & CARNAHAN, 1994][BELLOVIN & CHESWICK, 1994].

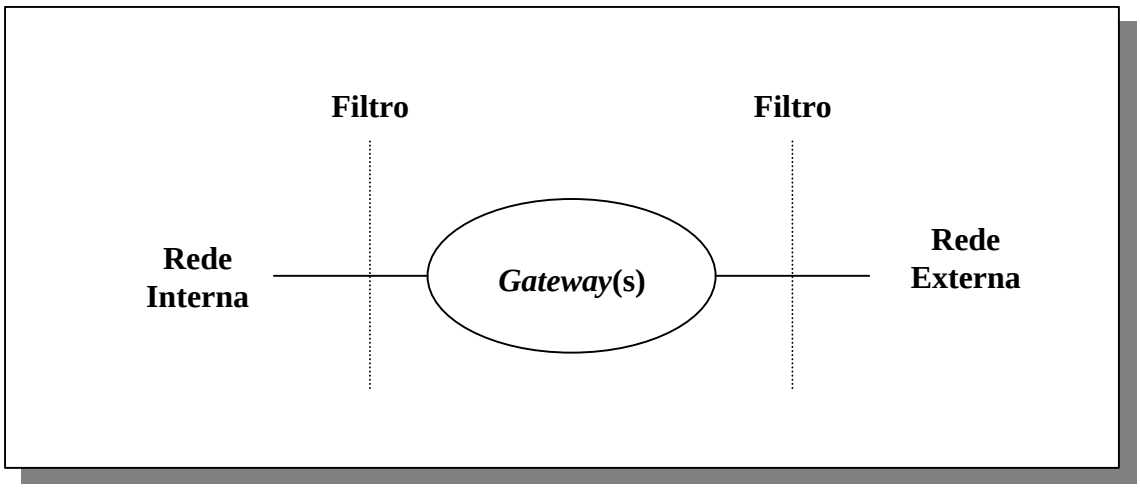

Figura 2-4 - Componentes de um Firewall [SOARES, 1995]

2.5 Política de Segurança

Uma política de segurança é um conjunto de leis, regras e práticas que regulam como uma organização gerencia, protege e distribui suas informações e recursos [SOARES, 1995].

Um dado sistema é considerado seguro em relação a uma política de segurança, caso garanta o cumprimento das leis, regras e práticas definidas nessa política.

Uma política de segurança deve incluir regras detalhadas definindo como as informações e recursos da organização devem ser manipulados ao longo de seu ciclo de vida, ou seja, desde o momento que passam a existir no contexto da organização até quando deixam de existir.

As regras que definem uma política de segurança são funções das designações de sensibilidade, associadas aos recursos e informações (por exemplo: não classificado, confidencial, secreto e ultra-secreto), do grau de autorização das entidades (indivíduos ou processos agindo sob o comando de indivíduos) e das formas de acesso suportadas por um sistema.

A implantação de uma política de segurança baseia-se na aplicação de regras que limitam o acesso de uma entidade às informações e recursos, com base na comparação do seu nível de autorização relativo a essa informação ou recurso, na designação da sensibilidade da informação ou recurso e na forma de acesso empregada. Assim, a política de segurança define o que é, e o

que não é permitido em termos de segurança, durante a operação de um dado sistema. A base política de segurança é a definição do comportamento autorizado para os indivíduos que interagem com um sistema [SOARES, 1995].

Uma quebra de segurança pode ser definida como uma ação, ou conjunto de ações, que vão contra 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. A política de segurança irá definir o que pode ou não pode ser feito, por quem, sob quais circunstâncias. Ela 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 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.

2.6 Sistemas de Detecção de Intrusão

Uma intrusão pode ser definida como: "*um conjunto de ações que tentam comprometer a integridade, confidencialidade ou disponibilidade de recursos"* [CROSBIE & SPAFFORD, 1995a].

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).

Intrusões são difíceis de detectar porque existem muitas formas pela qual elas podem acontecer. Intrusos podem explorar as fraquezas conhecidas da arquitetura dos sistemas ou explorar o conhecimentos de um sistema operacional para conseguir a autenticação normal de um processo. A tentativa de retirar uma falha do sistema pode introduzir uma nova falha ou expor uma falha existente, dando a oportunidade para um novo ataque.

Tentativas de ataque acontecem de acordo com algumas técnicas de acesso e freqüentemente o invasor estava fisicamente fora do sistema sob ataque. Hoje, percebemos que boa parte, senão a maioria dos ataques, partem internamente do próprio ambiente. 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.

2.6.1 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 significante da norma estabelecida é considerado anômalo. Isto é executado pela construção de um modelo estatístico que contém métricas derivadas do sistema operacional e

estabelece como intrusivo uma métrica observada que tem um desvio estatístico significante deste modelo.

- 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. Desta forma, a detecção é executada pela observação da exploração de pontos fracos conhecidos no sistema, o qual podem ser descritos por um padrão específico ou seqüência de eventos ou dados (uma "assinatura" da intrusão).

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 e, de preferência, em tempo real.

Em termos do tratamento dos dados, existem sistemas baseados em rede (*network based*) e sistemas baseados em *host* (*host based*). 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 em um único *host.*

Em ambos os casos, o sistema de detecção de intrusão (SDI) é freqüentemente um grande módulo monolítico. Este módulo, executa todo o monitoramento, obtenção de dados, manipulação destes dados e tomada de decisão para todo o sistema. Um problema observado neste tipo de abordagem para a construção de um SDI é o *overhead* imposto no sistema que está sendo protegido.

É importante ressaltar que um sistema de detecção de intrusão não inclui prevenção de intrusão, somente detecção e a comunicação de sua ocorrência ao administrador.

2.6.2 Características Desejáveis de um Sistema de Detecção de Intrusão

As características seguintes são identificadas como desejáveis em um Sistema de Detecção de Intrusão:

- O sistema deve estar em execução continuamente com um mínimo de supervisão humana;

- O sistema deve ser tolerante a falhas, de forma que deve recuperar-se quando o sistema é interrompido, de forma acidental ou causada por atividades maliciosas. Depois de reinicializado, o IDS deve retornar ao estado anterior e retomar às operações ao qual estava habilitado.

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

2.6.3 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*) [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, 1997].

2.6.4 Sistemas Baseados em *Data Mining*

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.

Lee e Stolfo [LEE & STOLFO, 1997] 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 2-5 apresenta a arquitetura sugerida.

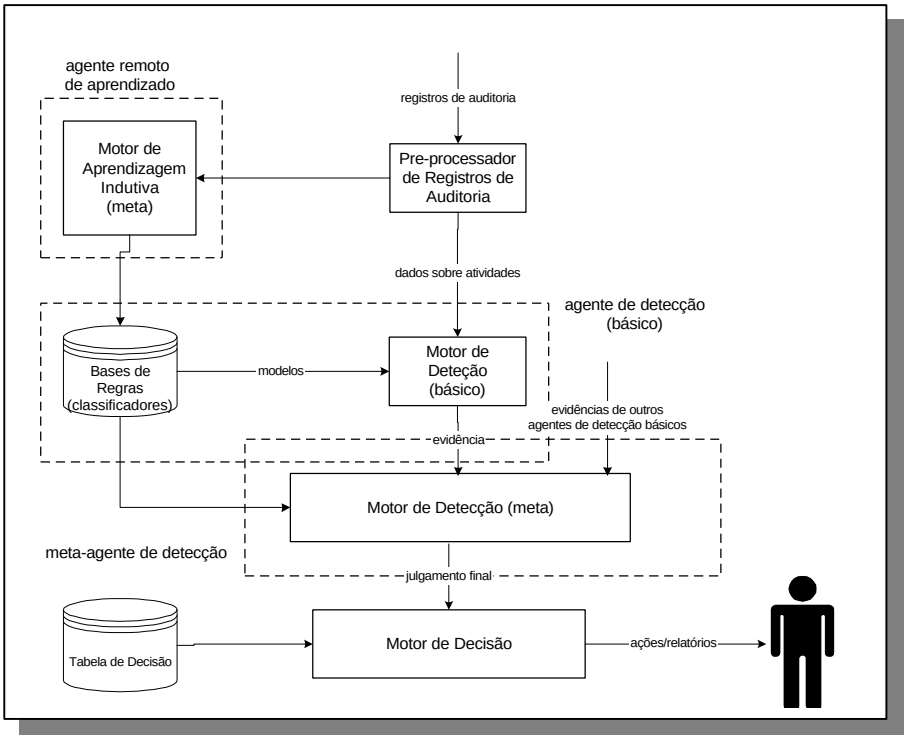

Figura 2-5 - Arquitetura de um Sistema de Detecção de Intrusão Baseado em *Data Mining*

2.6.5 O Sistema de Detecção de Intrusão do ICMC

Uma das características inovadoras deste sistema de detecção de intrusão consiste em introduzir um agente de segurança capaz de detectar comportamento intrusivo em conexões estabelecidas [CANSIAN et al., 1997a, 1997b, 1997c] [BONIFÁCIO Jr. et al., 1998a]. 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 é 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 2-6 mostra a arquitetura global do sistema.

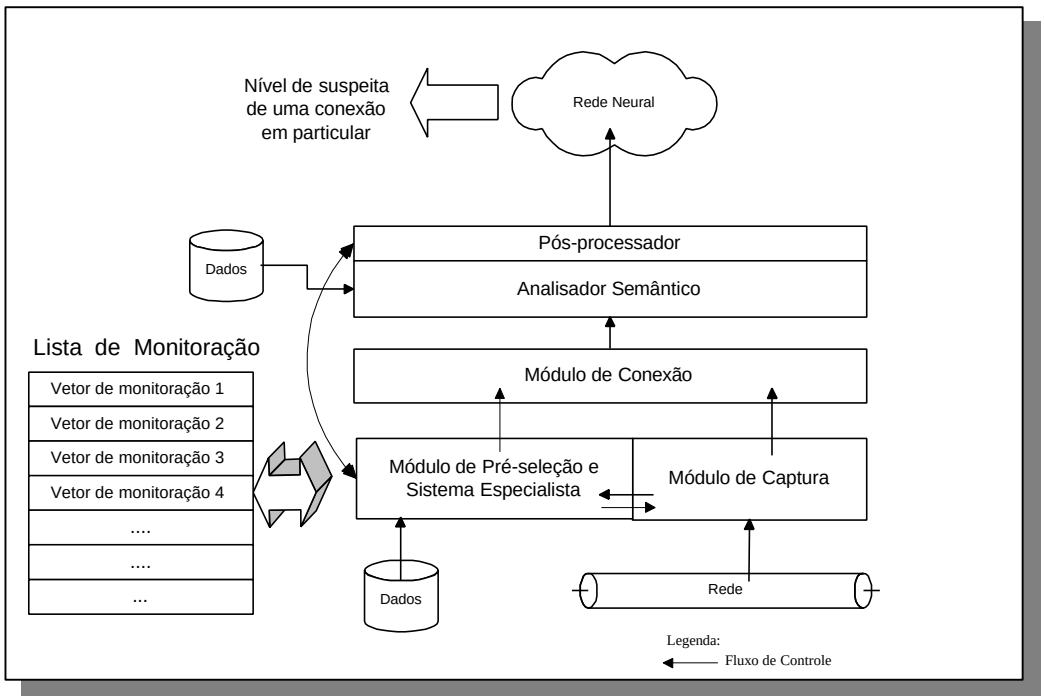

Figura 2-6 - Arquitetura global do sistema de detecção de intrusão do ICMC

2.7 Considerações Finais

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 (p.e.,detecção de intrusão e *firewall*) permite uma atuação mais consistente e eficaz na determinação dos problemas de segurança e na solução dos mesmos, seja ela manual ou automática.

Alguns exemplos de trabalhos nesta linha de pesquisa são apresentados por [BONIFÁCIO Jr. et al., 1998b] e [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].

Como visto, uma recente tecnologia de suporte à segurança em redes de computadores é a introdução do uso de Agentes Autônomos. O conceito de Agente e sua utilização no apoio a segurança são discutidos nos próximos capítulos.

3 Agentes

3.1 Considerações Iniciais

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, melhorar as possibilidades de reutilização e aumentar a qualidade de software. 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 [REAMI, 1998].

Agentes, *Agentes Inteligentes* e *sistemas baseados em agentes* têm atraído um considerável interesse de muitos campos da ciência da computação [LINGNAU & DROBNIK, 1996]. A tecnologia de agentes vem sendo aplicada academicamente nos mais diversos campos, principalmente em inteligência artificial, sistemas distribuídos e engenharia de software.

Recentemente algumas aplicações comerciais e muitas pesquisas acadêmicas vêm sendo desenvolvidas com esta tecnologia. Dentre as aplicações pode-se citar:

- 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];
- Busca de informações em bases de dados;
- Segurança Computacional [REAMI, 1998];
- Comércio Eletrônico [RODRIGUEZ, 1999];
- Computação móvel [LINGNAU & DROBNIK, 1996].

3.2 Definição de Agente

Infelizmente não existe um consenso dos pesquisadores sobre a exata definição de agente. Uma definição radicalmente baseada em Inteligência Artificial é apresentada por Selker: "*Agentes são programas de computadores que simulam o relacionamento humano, por fazerem exatamente o que uma outra pessoa faria por você"*[SELKER, 1994]. Minsk define um agente como *"um sistema que pode servir como um mensageiro, pelo motivo de possuir algumas habilidades de especialista"* [MINSK & RIECKEN, 1994]. Para Genesereth & Ketchpel agentes são "*componentes de software que comunicam com seus pares por troca de mensagens em uma linguagem de comunicação de agentes"*[GENESERETH & KETCHPEL, 1994].

O problema com a definição deve-se, em parte, à 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 (agente de viagem, agente econômico, agente de seguros, etc.) [REAMI, 1998]. Segundo o dicionário Aurélio da língua Portuguesa, o termo agente pode ser definido como "*aquele que trata de negócios por conta alheia*" [FERREIRA, 1977]. Segundo Nwana [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.

Em termos gerais, um agente pode ser definido como um software capaz de executar uma tarefa complexa em nome de um usuário [ENDLER, 1998].

Com tantas definições, torna-se interessante apresentar uma que seja extremamente abrangente e uma taxonomia para os diversos tipos de agentes e funções que executam. Franklin e Graesser após fazer várias comparações, definem o conceito de agente autônomo: *"Um agente autônomo é um sistema situado em um ambiente, e parte do mesmo, percebendo este ambiente e agindo sobre o mesmo no decorrer do tempo, de acordo com sua própria agenda e de modo a afetar o que ele perceberá no futuro"* [FRANKLIN & GRAESSER, 1996].

As várias definições existentes discutem um conjunto de propriedades para um agente. Como forma ajudar na classificação de um agente, Franklin e Graesser ainda apresentam um conjunto de propriedades que podem ser observadas na tabela abaixo:

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

Tabela 1 - Possíveis propriedades de agentes [FRANKLIN & GRAESSER, 1996]

Segundo a definição genérica apresentada por Franklin e Graesser, um agente autônomo sempre possui as quatro primeiras propriedades apresentadas na Tabela 1. Adicionar outras propriedades incrementam a aplicabilidade dos agentes. Outros autores, como Genesereth & Ketchpel, argumentam que agentes precisam necessariamente ser comunicativos: "*uma entidade é um software agente se e somente se ele comunica corretamente através de uma linguagem de comunicação de Agentes"* [GENESERETH & KETCHPEL, 1994].

Conforme observado em Reami [REAMI, 1998], 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 [JENNINGS & WOOLDRIDGE, 1995].

3.3 Tipologia dos Agentes

Esta seção apresenta uma tentativa de colocar os agentes existentes em diferentes classes de agentes, através de organização das tipologias de agentes. Uma tipologia refere-se ao estudo dos tipos de uma entidade. Em [NWANA, 1996] é apresentado várias dimensões para classificar os agentes existentes.

Primeiramente, agentes podem ser classificados por sua mobilidade, ou seja, por sua habilidade de mover através da rede de computadores. Desta forma, os 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, não possuindo o modelo de raciocínio simbólico.

A terceira dimensão envolve classificação de acordo com as diversas propriedades ideais e fundamentais que um agente deveria apresentar. Em seu trabalho, [NWANA, 1996] 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.

Em sua apresentação no mundo real e nas pesquisas, os agentes podem aparecer em várias das dimensões descritas anteriormente. Ainda em seu trabalho, [NWANA, 1996] reduz a combinação dessas várias dimensões, que não seriam possíveis de representação gráfica simples, a uma lista arbitrária, mas que cobre a maioria dos tipos de agentes em investigação atualmente. Desta forma, é identificado 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-intenções[1] 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

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

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 (*Altavista*, *InfoSeek*, *Lycos*, 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 (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.

Entretanto, observamos algumas aplicações que se utilizam de dois ou mais desses tipos de agentes. Estas aplicações são referenciadas como sistemas de agentes heterogêneos [NWANA, 1996] .

3.4 Agentes Móveis e Segurança Computacional

A idéia de executar computação cliente-servidor pela transmissão de programas executáveis entre clientes e servidores tem sido popularizada recentemente por pesquisadores e desenvolvedores interessados em serviços inteligentes de redes de computadores.

É apresentado a seguir, as principais características dos agentes móveis, uma comparação com outras tecnologias e a motivação para sua utilização em um ambiente de segurança computacional.

3.4.1 Definição de Agentes Móveis

Assim como na definição de agentes, são apresentadas diversas propostas para definição de agentes móveis. Um dos trabalhos mais citados em toda literatura pesquisada é o relatório de pesquisa apresentado por Chess, Harrison e Kershenbaum [CHESS et al, 1995]. Segundo eles, agentes móveis *"são programas tipicamente escritos em uma linguagem script, que podem ser disparados de um computador cliente e transportados para um computador remoto para execução"*. Além disso, esses elementos possuem características inerentes ao conceito de multiagentes, que proporcionam um bom desempenho em sistemas de objetos distribuídos. Assim, características como cooperação, autonomia e representatividade foram herdadas da sua própria origem, além de serem acopladas outras tantas a fim de suprir as necessidades exigidas para o bom funcionamento de modelos que utilizam esse paradigma, a saber:

Objetos passantes: quando um agente móvel é transferido, todo o objeto é movido, ou seja, o código, os dados, o itinerário para chegar ao servidor necessário, o estado de execução, etc.

Assincronismo: o agente móvel possui sua própria *thread* de execução e esse não precisa ser executado sincronamente.

Interação local: o agente móvel interage com outros agentes móveis ou objetos estacionários locais. Se necessário, um agente mensageiro é despachado para facilitar a interação com agentes remotos.

Operações sem conexão: o agente móvel pode executar tarefas mesmo com a conexão fechada. Quando se faz necessário transferência de agentes, aguarda-se até que a conexão seja restabelecida.

Execução paralela: múltiplos agentes podem ser disparados para diferentes servidores a fim de executar tarefas em paralelo.

O modelo básico de um sistema que utiliza o paradigma de agentes móveis está esquematizado na Figura 3-1. A aplicação cliente nada mais é que um ambiente de interação com o usuário, que utilize os conceitos de agentes móveis. Além disso, essa aplicação tem interface com o ambiente agente. Logo, quando as tarefas são executadas, o ambiente de execução de agente envia o agente móvel, através do subsistema de mensagens.

No servidor, ocorre a ação contrária: o agente é recebido através do subsistema de mensagem e é encaminhado ao ambiente de execução de agente, que deve ser compatível com o ambiente do cliente. Neste ponto, as tarefas são realizadas a nível de aplicação servidora e se necessário, um agente mensageiro é disparado.

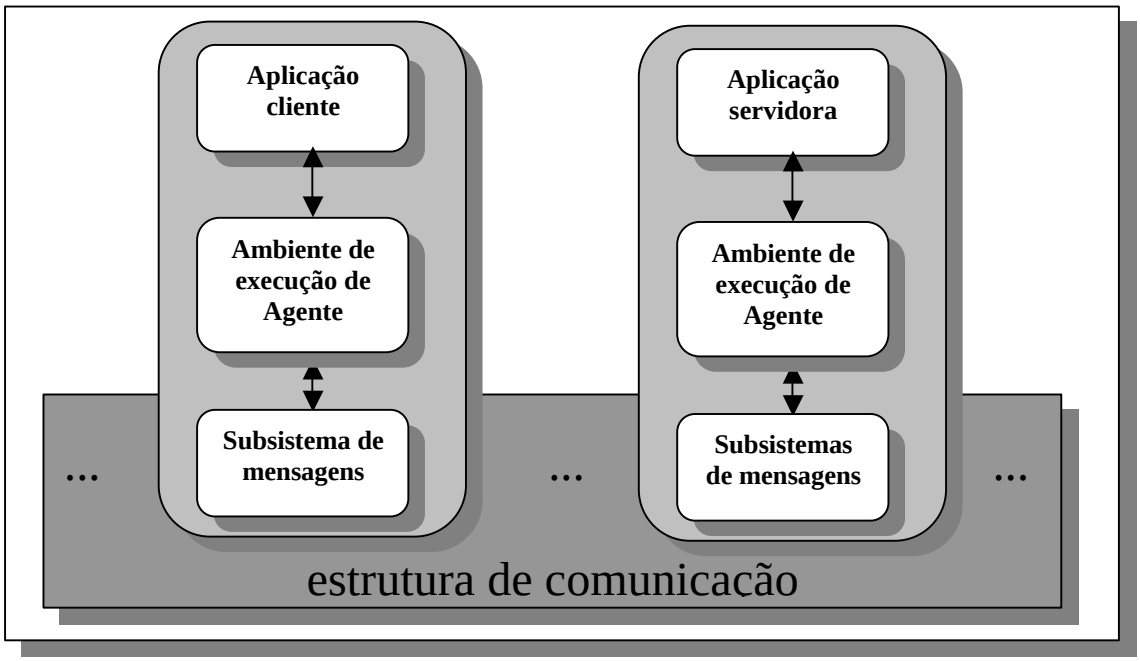

Figura 3-1: Modelo básico de um sistema que utiliza agentes móveis [CHESS et al, 1995]

3.4.2 Arquiteturas Cliente/Servidor

Em virtude das muitas definições encontradas na literatura apresentarem o conceito de que um agente móvel é um programa que pode ser disparado de um computador cliente e transportado para um computador remoto para execução, torna-se necessário diferenciá-los dos demais mecanismos de comunicação entre processos. Nesta seção são apresentados três mecanismos bastante conhecidos dos programadores em geral.

3.4.2.1 RPC – Remote Procedure Call

O mecanismo de RPC permite que programas chamem procedimentos localizados em outras máquinas. Quando um processo rodando em uma máquina *A* chama um procedimento em uma máquina B, o processo que chamou é suspenso, e a execução do procedimento é levada a efeito na máquina *B*. A informação pode ser transportada do processo que chama para o procedimento chamado através de parâmetros, e voltar para o processo como resultado da execução do procedimento.

O modelo para RPC é esquematizado na Figura 3-2. Apesar de ser bastante utilizado, ele apresenta alguns problemas relacionados, sobretudo, à sua origem ligada a transações em redes locais, não apresentando características robustas exigidas por uma rede geograficamente distribuída.

Quando ocorre a identificação de uma chamada RPC durante a execução de uma transação no cliente, é sabido que o servidor será consultado. Então, no *stub* do cliente os parâmetros são empacotados e enviados, através do *kernel*, para o servidor. Chegando ao *stub* do servidor, os parâmetros são desempacotados e o servidor realiza as tarefas necessárias. Depois, o servidor retorna para o *stub*, onde os resultados serão empacotados e reenviados ao cliente, através do kernel.

De volta ao *stub* do cliente, os resultados são desempacotados e disponibilizados.

Alguns pontos das características internas dos dois modelos, agentes móveis e RPC, são avaliados e comparados a seguir.

Primeiramente, é fácil perceber a diferença com relação à robustez dos dois mecanismos. Enquanto o RPC foi desenvolvido para redes LAN, o paradigma dos agentes móveis foi especificado para suportar características dos sistemas distribuídos atuais. Ou seja, é mais simples acoplar mecanismos de tolerância a falhas, segurança, etc., em um modelo que já foi desenvolvido para a implementação de tais características.

Outra diferença é observada no que diz respeito ao tráfego na rede. No RPC há um maior número de chamadas por transação, enquanto que a execução de uma transação pode ser realizada por um único agente móvel. Como conseqüência, o tráfego na rede de um sistema que utiliza o modelo RPC é maior, apesar do aumento instantâneo de tráfego, na ocorrência da transferência de um agente. A Figura 3-3 apresenta graficamente a diferença na quantidade de comunicações entre um modelo de agentes móveis e um modelo cliente/servidor. Apesar disso, no caso do modelo RPC, o cliente normalmente, espera um retorno do servidor para continuar uma transação, enquanto que, no segundo caso, um agente é enviado com toda a tarefa realizada, estando o cliente livre para realização de outras tarefas.

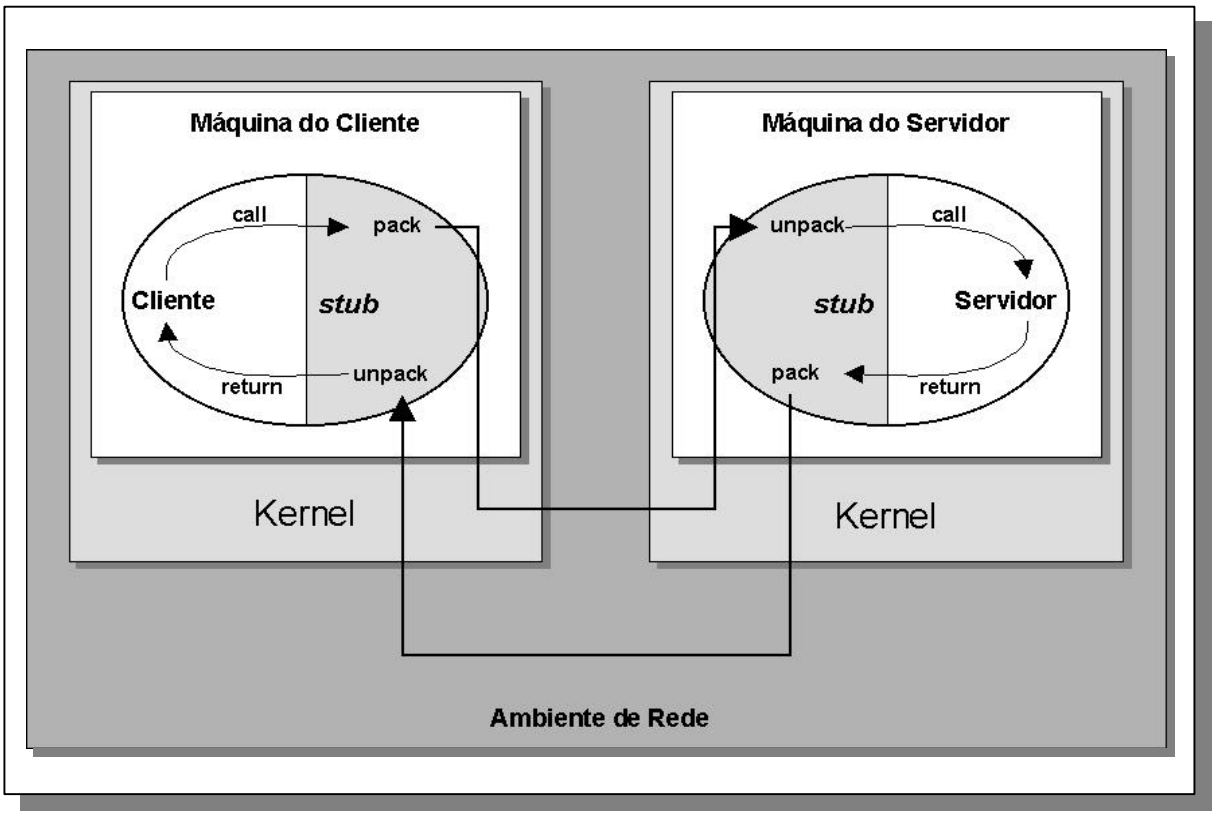

Figura 3-2: Mecanismo de RPC [TANENBAUM, 1997].

Como comentado anteriormente, o RPC é um modelo síncrono, ou seja, o cliente e o servidor devem fazer suas transações em total sincronismo um com o outro. No caso dos agentes, isso não é verdade. Um cliente, quando dispara um agente móvel para o servidor, está livre para atuar em outro agente ou outra tarefa desejada.

Além disso, na ocorrência de falha de conexão, a transação RPC estará perdida caso não haja um mecanismo que garanta a persistência do estado de execução.

Tal mecanismo pode ser anexado ao modelo, porém com um custo em *overhead* de desempenho. Enquanto isso, os agentes móveis não precisam passar por esse tipo de problema, pois o cliente é capaz de realizar as tarefas correspondentes sem conexão, disparando o agente apenas quando contém toda a transação. Além disso, o próprio agente móvel possui seu estado de execução evitando perdas no caso desse tipo de falha.

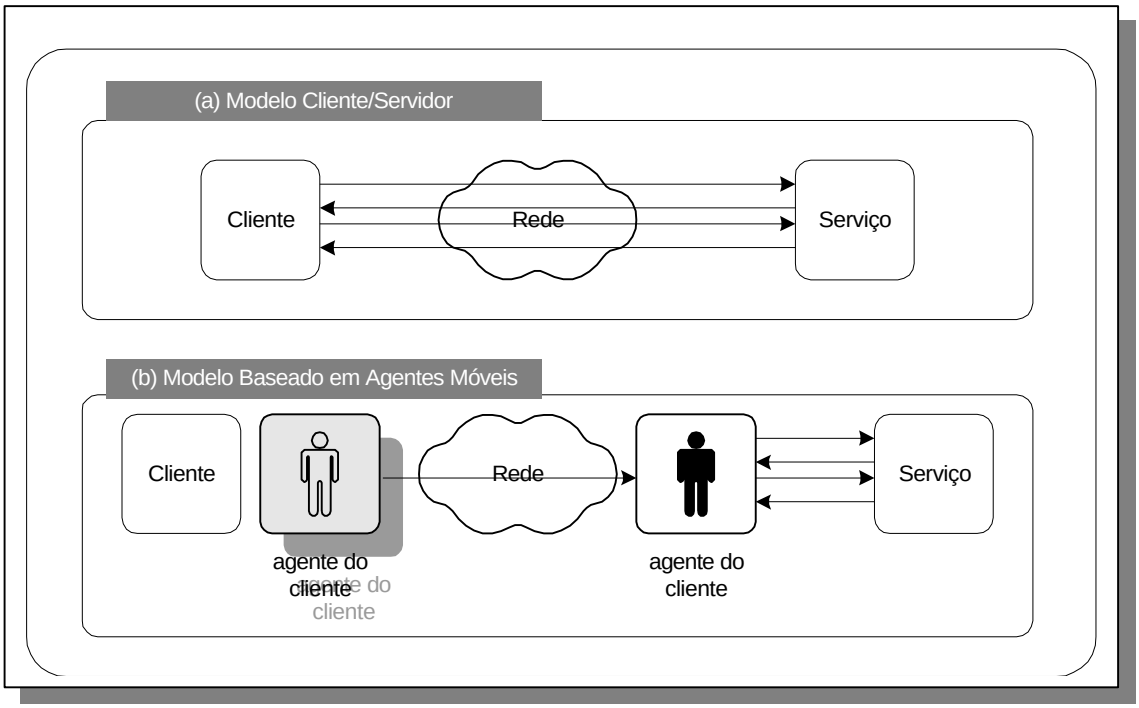

Figura 3-3 Contraste entre o Modelo Cliente/Servidor e o Modelo Baseado em Agentes Móveis [REAMI, 1998]

3.4.2.2 Ambientes de Passagem de Mensagem

Um ambiente de passagem de mensagem consiste basicamente em uma biblioteca de comunicação que, atuando como uma extensão das linguagens seqüenciais (como C e Fortran), permite a elaboração de aplicações paralelas.

Os ambientes de passagem de mensagens não foram desenvolvidos especificamente com o intuito de utilizar os sistemas distribuídos para o desenvolvimento de aplicações paralelas. Eles foram desenvolvidos inicialmente para máquinas com processamento maciçamente paralelo (*Massively Parallel Processing* - MPP) onde, devido à ausência de um padrão, cada fabricante desenvolveu seu próprio ambiente, sem se preocupar com portabilidade. Com o passar dos anos, muita experiência foi adquirida, pois os diferentes projetos de interfaces de passagem de mensagens enfatizavam aspectos diferentes para o seu sistema. Exemplos desses sistemas são *nCUBE* PSE, *IBM EUI*, *Meiko CS System* e *Thinking Machines CMMD*.

Com o objetivo de acabar com o problema de portabilidade, vários grupos de pesquisa desenvolveram ambientes de passagem de mensagens com plataforma portátil. A idéia é definir um conjunto de funções independentes da máquina que está sendo utilizada e implementá-las em várias plataformas de hardware.

As aplicações puderam, com isso, ganhar a portabilidade perdida e serem executadas em todos os equipamentos para os quais o ambiente foi desenvolvido. Recentemente os ambientes portáteis também têm sido desenvolvidos para sistemas heterogêneos, onde dois ou mais tipos de computadores diferentes cooperam para resolver um problema. Exemplos de plataformas portáteis para equipamentos heterogêneos são: P4, PARMACS, Express, PVM, MPI, entre outros, sendo o PVM e o MPI os mais conhecidos.

PVM – *Parallell Virtual Machine*

O PVM é um conjunto integrado de bibliotecas e de ferramentas de software, cuja finalidade é emular um sistema computacional concorrente heterogêneo, flexível e de propósito geral [BEGUELIN, 1994].

Diferente de outros ambientes portáteis desenvolvidos inicialmente para máquinas com multiprocessadores (como o P4, Express e outros), o PVM nasceu com o objetivo de permitir que um grupo de computadores interconectados, possivelmente com diferentes arquiteturas, possa trabalhar cooperativamente formando uma máquina paralela virtual.

O sistema PVM é composto por duas partes [GEIST, 1994]. A primeira é um *daemon*, chamado pvmd3 (ou simplesmente pvmd), que reside em todos os *hosts*, compondo a máquina virtual. O termo máquina virtual será utilizado para designar um computador lógico com memória distribuída e o termo *host* será para designar um dos computadores que formam a máquina virtual.

O pvmd foi projetado para ser instalado na máquina por qualquer usuário com um *login* válido. Quando um usuário deseja executar uma aplicação PVM, ele primeiro deve criar a máquina virtual iniciando o PVM. A aplicação pode então ser iniciada a partir do *prompt* do sistema operacional (normalmente, mas não necessariamente o UNIX), em qualquer computador

pertencente à máquina virtual. Pode haver mais de uma máquina virtual utilizando os mesmos equipamentos simultaneamente, sendo que cada aplicação não interfere nas demais.

A segunda parte do sistema é uma biblioteca de rotinas da interface PVM. Ela contém um conjunto completo de primitivas que são necessárias para a cooperação entre as tarefas de uma aplicação. Essa biblioteca contém rotinas que podem ser chamadas pelo usuário para a passagem de mensagens, geração de processos, coordenação de tarefas e modificação da máquina virtual.

O sistema PVM permite que sejam escritas aplicações nas linguagens Fortran, C e C++. A escolha por esse conjunto de linguagens deve-se ao fato de que a maioria das aplicações passíveis de paralelização está escrita nessas linguagens.

Através do PVM, uma coleção de computadores heterogêneos (seriais, paralelos e vetoriais) desempenha as funções de um computador com memória distribuída e com alto desempenho. O PVM fornece as funções que iniciam automaticamente as tarefas (*tasks*) na máquina virtual e permitem a comunicação e a sincronização entre elas. Uma tarefa é definida como uma unidade computacional em PVM análoga aos processos UNIX (freqüentemente, não necessariamente, é um processo UNIX).

O modelo computacional do PVM é, portanto, baseado na noção de que uma aplicação consiste de várias tarefas. Cada tarefa é responsável por uma parte da carga de trabalho da aplicação.

Uma aplicação pode ser paralelizada por dois métodos: o paralelismo funcional e o paralelismo de dados. No paralelismo funcional (também conhecido como paradigma mestre-escravo) a aplicação é dividida através das suas funções, isto é, cada tarefa desempenha um serviço diferente, como por exemplo, entrada, processamento e saída.

O paralelismo de dados refere-se ao paradigma SPMD (*Single Program – Multiple Data*), que consiste de um único programa sendo executado em todos os processadores de uma máquina MIMD (*Multiple Instruction Stream Multiple Data Stream)*, mas com diferentes dados e com possíveis execuções diferentes em cada processador. O PVM permite qualquer um dos métodos, como também um método híbrido (uma mistura dos dois). A Figura 3-4 mostra um exemplo do modelo computacional do PVM e uma visão arquitetural destacando a heterogeneidade do sistema.

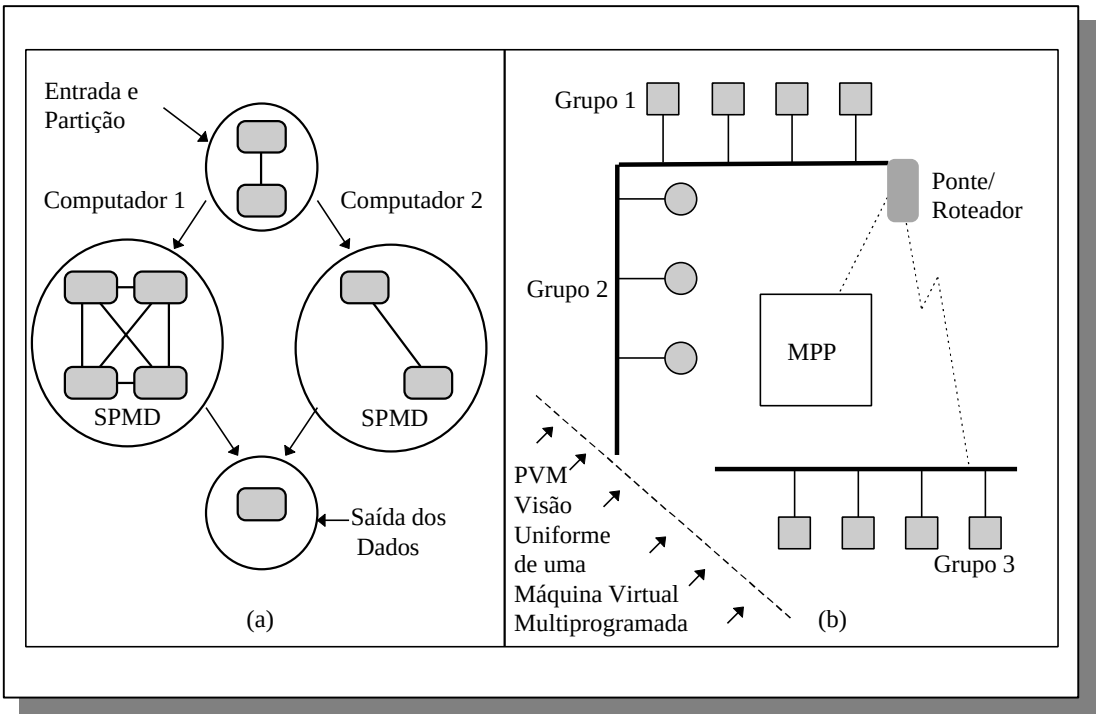

Figura 3-4 - O PVM. (a) Modelo Computacional (b) Visão Arquitetural

MPI – *Message Passing Interface*

O MPI é um padrão de interface de passagem de mensagens para aplicações que utilizam computadores MIMD com memória distribuída. Ele não oferece nenhum suporte para tolerância a falhas e assume a existência de comunicações confiáveis. O MPI não é um ambiente completo para programação concorrente, visto que ele não implementa: I/O paralelos, depuração de programas concorrentes, canais virtuais para comunicação e outras características próprias de tais ambientes [WALKER, 1994].

Um conjunto de rotinas responsáveis pela comunicação ponto-a-ponto entre os pares de processos forma o núcleo do MPI. São implementadas rotinas bloqueantes e não bloqueantes para enviar e receber mensagens.

A rotina que envia a mensagem no modo bloqueante não retorna enquanto a mensagem contida no *buffer* não estiver segura. No modo não bloqueante a rotina que envia a mensagem pode retornar enquanto a mensagem ainda está “volátil”.

Para receber uma mensagem no modo bloqueante, a rotina não retorna até que a mensagem seja inserida no *buffer*. No modo não bloqueante a rotina pode retornar antes da mensagem ter chegado.

MPI possui grupos de processos e rotinas para o gerenciamento dos grupos. Os grupos podem ser usados para duas funções distintas. Na primeira os grupos especificam os processos envolvidos em uma operação de comunicação coletiva, como um *broadcasting.* Toda a comunicação ocorre dentro e através dos grupos. Na segunda eles podem ser usados para introduzir o paralelismo dentro da aplicação, onde diferentes grupos realizam diferentes tarefas.

Cada grupo pode possuir códigos executáveis diferentes ou o mesmo código.

3.4.2.3 Comparações

Uma característica importante de agentes móveis, que os diferenciam das tecnologias apresentadas anteriormente, é que eles não estão confinados no sistema onde começa sua execução. Agentes móveis são livres para serem executados em qualquer máquina do ambiente ao qual estão inseridos. Ao chegar em um ponto remoto, o Agente pode tomar a decisão (autonomia) de migrar para um novo computador, seguindo sua lista de itinerários. Como visto, agentes móveis possuem a habilidade de transportar a si próprios (código, estado de execução, itinerário, etc) de um sistema para outro numa rede de computadores. Esta habilidade de viajar permite a um agente móvel se movimentar para o sistema de agente que contém o objeto com o qual deseja interagir. O processo disparado com os demais mecanismos está sempre subordinado a um processo de origem.

3.4.3 Aplicação para Agentes Móveis

Muitos autores têm sugerido que agentes móveis oferecem um importante e novo método para execução de transações e recuperação de informações em redes de computadores. Outros apontam que agentes móveis podem ser uma importante tecnologia para apoio a segurança computacional.

Segundo [LINGNAU & DROBNIK, 1996], agentes móveis constituem uma nova abordagem para a arquitetura e implementação de sistemas Distribuídos. Um agente móvel é um programa com uma identidade persistente que move através de redes de computadores e podem comunicar com o ambiente e outros agentes. Possíveis aplicações para agentes móveis incluem gerenciamento de rede, recuperação de informação, simulação distribuída, controle de dispositivos remotos, documentos ativos e computação móvel.

Em [RODRIGUEZ, 1999] é apresentado uma modelagem para Comércio Eletrônico usando CORBA que consiste de um agente móvel que procura produtos numa rede em nome de um cliente.

3.4.4 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:

- *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 et. al., 1997]. A Figura 3-5 mostra a arquitetura de um sistema de agentes móveis.

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.

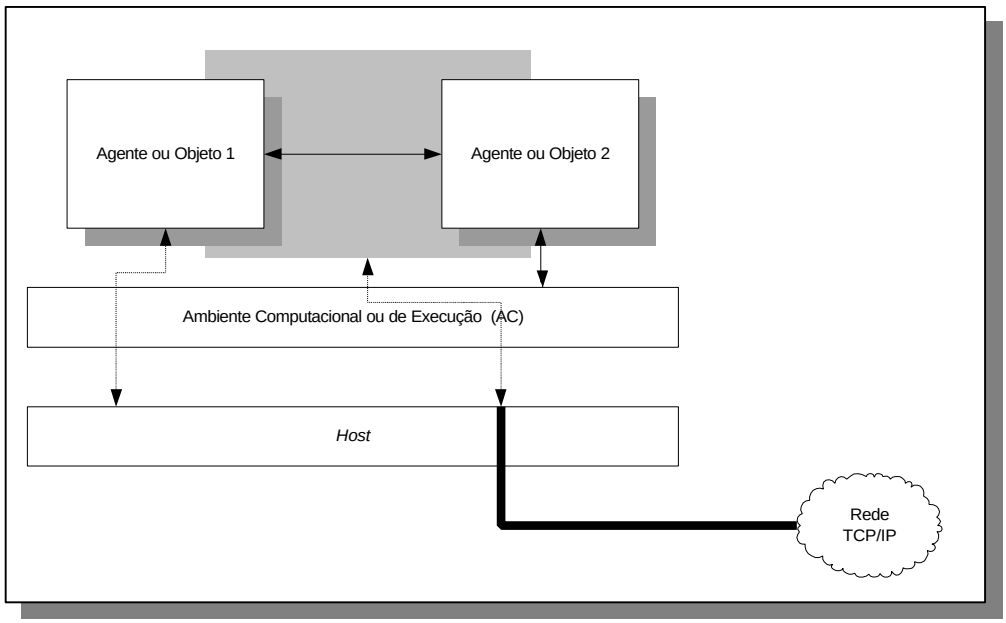

Figura 3-5 - Arquitetura Típica de um Sistema de Agentes ou Objetos Móveis [VITEK, 1997]

3.4.5 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.

Conforme observado em [REAMI, 1998], Sander e Tschudin [SANDER & TSCHUDIN, 1998] fornecem um exemplo interessante do tipo de problema de segurança que pode ocorrer durante uma busca por preços de passagem feita por um agente de compras. Suponha que o objetivo programado pelo usuário seja visitar os servidores de várias empresas aéreas, encontrar um vôo disponível e, uma vez determinada a melhor oferta, agendar o vôo. 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 vôo e, portanto, precisa assinar digitalmente através de sua chave-privada: a chave-privada é roubada.

Observando a Figura 3-5, é possível verificar as diversas fronteiras em que ocorrem os problemas de segurança relacionados com os sistemas de agentes. Vitek, Serrano e Thanos [VITEK et. al., 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 et. al., 1997]

3.4.6 Gerenciamento de Segurança Apoiado por Agentes Móveis

O objetivo desta seção é apresentar algumas justificativas para utilização de um ambiente baseado em agentes móveis em auxílio ao gerenciamento de segurança de uma redes de computadores.

3.4.6.1 Intrusões Móveis

Visto que boa parte das violações de segurança são violações internas, intencionais ou não, partindo muitas vezes dos próprios funcionários ou pessoas relacionadas ao sistema, torna-se necessária a utilização de técnicas que não se limitam a monitorar apenas a "porta de conexão" com mundo externo, como é o caso dos mecanismos de *firewall*.

Técnicas como a análise direta dos pacotes trafegando pela rede e a sua comparação com padrões estáticos definidos, também são bastante utilizadas. A inclusão de inteligência artificial, ou mais especificamente de sistemas baseados em regras, de redes neurais e de técnicas de análise e extração de informações de banco de dados como *data mining* têm sido constantes nos sistemas atuais.

Entretanto, os sistemas que realizam a monitoração de segurança freqüentemente também são alvo 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, ou seja, por reinicio do processo do sistema de monitoração.

Desta forma, uma maneira de contornar os problemas apresentados anteriormente é a 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 e de facilidades para a definição de tarefas executadas pelos agentes.

3.4.6.2 Apoio ao Gerenciamento de Segurança

Em um processo de gerenciamento de segurança observamos a dependência de um processo contínuo de melhorias e conseqüentemente, 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 as formas de segurança, implementação de tais formas, avaliação constante e correções**.**

Este processo contínuo demanda muita atenção e tempo do gerente de segurança. Entretanto, alguns agravantes têm contribuído para insucesso das tentativas de gerência da segurança em muitos ambientes:

- O número de redes de computadores existentes está aumentando muito e, portanto, a demanda por profissionais capacitados para executar o gerenciamento de segurança também aumenta;

- 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 nas mais diversas redes.

Levando em consideração tais fatores, fica clara a necessidade de um ambiente que possa fornecer apoio ao profissional responsável pela gerência 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 característica de autonomia dos agentes de software pode fornecer soluções interessantes para 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.

3.4.7 Uma Metáfora para o Ambiente

Em seu trabalho, [REAMI, 1998] apresentou uma metáfora extremamente interessante para o sistema de gerenciamento de segurança apoiado por agentes móveis. Inspirado na tradição medieval japonesa, sugeriu a utilização das 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".

Na analogia apresentada, o *Shogun* seria um sistema gerente e 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.

Reami ainda apresenta a possibilidade de utilização de outras metáforas. Por exemplo, temos o sistema de defesa do organismo, no qual células especializadas realizam o trabalho de destruição de elementos considerados estranhos, ou ainda, sociedades de insetos que como abelhas, formigas e cupins, nas quais indivíduos especializados realizam a tarefa de proteção dos seus membros e recursos, etc.

No desenvolvimento deste trabalho, será utilizada a metáfora do Shogun e dos Samurais, visto que esta se tornou bastante interessante e muito semelhante à idéia do ambiente proposto.

3.5 Considerações Finais

O termo agente tem sido empregado de diversas formas no mundo da computação e especialmente no da inteligência artificial. Apesar de existirem diversas definições para o termo

agente, estas estão convergindo para o conceito de que algumas características fundamentais são necessárias. Principalmente as características de reatividade, autonomia, orientação à metas e continuidade.

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 [REAMI, 1998].

Com a crescente utilização das redes de computadores e conseqüente intensificação das mais diversas formas de intrusão, a tarefa de gerenciamento de segurança torna-se cada vez mais complexa. Fica evidente a necessidade de uma ambiente de segurança que possa fornecer apoio ao gerente de segurança. Tal ambiente deverá apresentar a capacidade de mobilidade pela rede em busca de comportamentos intrusivos, ter autonomia para agir em nome do gerente de segurança e ainda, tomar decisões frente algum comportamento caracterizado como intrusivo.

Como visto, a tecnologia de agentes, mas especificamente agentes móveis, fornece os subsídios para o desenvolvimento de tal ambiente.

A proposta para um Sistema de Detecção de Intrusão (SDI) baseado em agentes móveis é modelada no próximo capítulo.

4 Proposta para um SDI Baseado em Agentes Autônomos e Móveis

4.1 Considerações Iniciais

O grau de proteção a cada ação maliciosa está diretamente relacionado ao tempo e esforços gastos construindo e gerenciando os sistemas de segurança. Utilizando-se complexas ferramentas, as quais continuamente monitoram e notificam atividades consideradas suspeitas, pode-se conseguir identificar essas atividades no momento em que elas ocorrem. Entretanto, isso envolve um alto custo em termos de tempo e dinheiro na construção e gerenciamento de sistemas de monitoria. Esses sistemas também impõem penalidades de performance no ambiente que está sendo protegido, o que pode ocasionar em sua rejeição pelos usuários.

A arquitetura monolítica de Sistemas de Detecção de Intrusão (SDI), comumente utilizada em sistemas comerciais ou de pesquisa, apresenta um número de problemas que limitam sua capacidade de configuração, escalabilidade ou eficiência.

Este capítulo apresenta uma modelagem para o desenvolvimento de um Sistema de Detecção de Intrusão não-monolítico baseado em agentes móveis.

4.2 Vantagens de um SDI não-monolítico

A abordagem monolítica apresenta alguns problemas práticos. Se uma nova forma de intrusão não prevista no sistema é descoberta, o SDI deve ser completamente reconstruído para conseguir tratá-la e isso, com certeza, não é uma ação trivial. [ZAMBONI et al., 1998] [CROSBIE & SPAFFORD, 1995a, 1995b]

Outra preocupação diz respeito à tolerância a falhas, uma vez que um sistema monolítico apresenta-se como um único ponto de falha e ataques. Consequentemente, metodologias de ataques bem conhecidas (como por exemplo, ataques *Denial of Service*), quando proferidas contra a máquina que hospeda o SDI, comprometem por completo a integridade do sistema.

A utilização de agentes autônomos têm sido proposta por alguns autores como uma forma de se construir sistemas de detecção de intrusão não-monolíticos [CROSBIE & SPAFFORD, 1995a, 1995b] [ZAMBONI et al, 1998]. A capacidade dos agentes autônomos de manter informação específica do seu domínio de aplicação, nesse caso, aplicação de segurança, dá a estes agentes e consequentemente a todo o sistema, grande flexibilidade.

Em vez de um grande módulo monolítico, este trabalho apresenta a proposta de uma abordagem modular baseada em agentes autônomos e móveis para o desenvolvimento de um SDI. Este SDI consiste de um conjunto de pequenos processos (agentes) que podem agir independentemente no ambiente em construção. Eles serão desenvolvidos para moverem-se pelo ambiente no qual estão inseridos, observarem os comportamentos do sistema, cooperarem uns com os outros via passagem de mensagens, notificarem quando uma ação for considerada suspeita e ainda, proverem ações reativas (contra-ataque).

Cada agente observa somente um pequeno aspecto de todo o sistema. Um simples agente, sozinho, não pode formar um sistema de detecção de intrusão, uma vez que sua visão é limitada a uma pequena "fatia" do sistema. Entretanto, se muitos agentes operam em um sistema e cooperam uns com os outros, então um poderoso SDI pode ser desenvolvido. Uma vez que os agentes são independentes uns dos outros, eles podem ser adicionados e removidos do sistema dinamicamente, de forma que não é necessário reconstruir todo o SDI ou ainda, interromper suas atividades. Assim, a qualquer sinal de identificação de uma nova forma de ataque, novos agentes especializados podem ser desenvolvidos, adicionados ao sistema e configurados para atender uma política de segurança específica.

Outra vantagem da abordagem descrita anteriormente, é a facilidade de configuração apresentada pelo sistema em atendimento às necessidades políticas do ambiente ao qual está inserido. Isso torna-se uma característica importante uma vez que, conforme visto no capítulo 2, o que é considerado uma quebra de segurança para um ambiente pode não ser em outro, em função do tipo de informação que se quer proteger.

Uma vez que a mudança é subjacente a todo trabalho de software e que esta é inevitável quando se constrói sistemas baseados em computador, outra vantagem deste sistema é a sua alta manutenibilidade. Definida qualitativamente em [PRESSMAN, 1995] como sendo "a facilidade

com que um software pode ser entendido, corrigido e/ou aumentado", esta torna-se a meta primordial que orienta os passos de um processo de engenharia de um software.

Sendo dividido em módulos contendo um conjunto de pequenos agentes especializados em uma única função e que consequentemente apresentam uma menor complexidade lógica, o sistema procura minimizar o esforço gasto com a manutenibilidade. Isso deve-se ao fato de que, desta forma, cada agente apresenta uma estrutura bem compreensível, facilitando o seu entendimento e conseqüente necessidades de manutenção. Isso é refletido diretamente em termos de:

- Tempo de reconhecimento do problema;
- Tempo de análise do problema;
- Tempo de especificação das mudanças;
- Tempo de correção (ou modificação) ativa;
- Tempo de testes locais;
- Tempo de testes globais;
- Tempo de revisão de manutenção;
- Tempo de recuperação total.

Cada uma das métricas anteriores pode, de fato, ser registrada sem grandes dificuldades. Além dessas medidas orientadas para o tempo, a manutenibilidade pode ser medida indiretamente ao considerarmos as medidas da estrutura do projeto e as métricas da complexidade do sistema, que indicarão também um ganho significativo no momento da inserção de novas funções (novos agentes).

Além dessas vantagens, [CROSBIE & SPAFFORD, 1995a; 1995b] especificam um sistema baseado em agentes autônomos no qual as capacidades dos agentes são modificadas através do uso algoritmos genéticos. Os autores reconhecem, entre outras, as seguintes vantagens de sistemas baseados em agentes autônomos sobre sistemas monolíticos:

- **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 de maneira a gerar a menor sobrecarga possível no sistema;

- **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 possam ser mais úteis;

- **Resistência à subversão:** caso um sistema de defesa seja subvertido, ele poderá dar a falsa sensação de segurança. Entretanto, isto torna-se mais difícil, pois os conhecimentos adquiridos de um agente não dá o conhecimento das operações de outros, visto que eles desempenham funções diferentes.

- **Escalabilidade:** para atuar em sistemas maiores, basta adicionar mais agentes e aumentar sua diversidade.

4.3 Arquitetura do Sistema Proposto

O conceito principal que envolve o SDI baseado em agentes autônomos e móveis é a simplicidade. Cada agente é uma entidade simples que irá desempenhar uma atividade específica e cooperar com outros agentes de forma mais eficiente possível. Quando uma atividade for considerada suspeita por um agente, ele irá comunicar aos demais agentes do sistema sua suspeita de uma possível intrusão. Neste momento, será acionado um agente (ou um conjunto de agentes) com um maior grau de especialização naquele tipo de suspeita.

Naturalmente um agente poderá cometer um erro, que será identificado por um agente com um nível de especialização superior. Uma vez que um número maior de agentes suspeitam de uma possível intrusão, uma mensagem pode ser enviada pedindo a intervenção de um operador humano (via alguns processos de monitoramento) e agentes de reação poderão ser acionados.

Isso demonstra que uma decisão deverá ser tomada em conjunto. Nenhum agente possui a autoridade de, sozinho, identificar uma intrusão. Essa decisão será tomada com base no consenso de vários agentes no sistema. Se somente um agente suspeita de uma intrusão, ele poderá ser ignorado após uma votação dos agentes envolvidos naquela suspeita. Entretanto, se mais de um agente suspeitam de um comportamento anômalo, então há uma maior probabilidade de ser uma intrusão potencial e neste caso, poderá ser tomada a decisão de comunicar um operador humano ou acionar agentes especializados em contra ataque. Fica claro que certos eventos podem ser mais "importantes" neste esquema do que outros. Por exemplo, 50 falhas em tentativas de *login* como *root* receberá um grau de suspeita maior que uma conexão de *FTP* externa ao domínio monitorado.

A Figura 4.1 apresenta uma arquitetura (modelo em camadas) para o SDI proposto. As camadas são numeradas a partir da camada de Vigilância (camada 1), e cada uma delas representa um grupo de tarefas específicas desempenhas por agentes especializados nas funções desta camada. Através do mecanismo de troca de mensagens, um agente em uma camada aciona um ou mais agentes em uma camada superior. Em outras palavras, a camada N utiliza os serviços da camada N-1, desempenha suas funções e fornece serviços para a camada N+1.

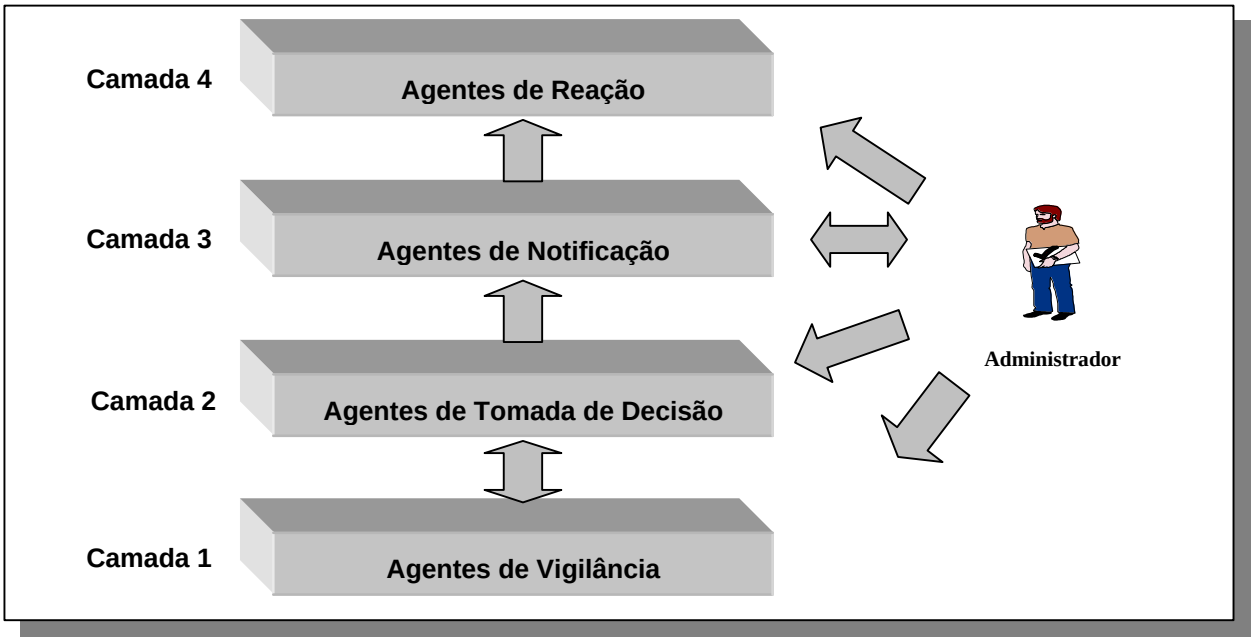

Figura 4-1 - Modelagem em Camadas para o Sistema Proposto

Com base em informações coletadas pelos *Agentes de Vigilância*, *Agentes de Tomada* de Decisão entrarão em ação, analisando e identificando possíveis intrusões. Caso uma ação seja

considerada suspeita por estes agentes, *Agentes de Notificação* serão acionados e cuidarão de notificar o administrador da rede (via *e-mail*, *pager*, chamada telefônica, alarme, etc.) ou acionar os agentes de nível superior. Em último nível, encontram-se os *Agentes de Reação*. Estes agentes, cuidarão de contra-atacar automaticamente as possíveis intrusões, com base nas informações dos agentes de notificação ou ainda, serem acionados através de uma intervenção do administrador da rede.

Apesar do cenário anterior exemplificar uma comunicação *bottom-up* através das camadas da arquitetura proposta, há a possibilidade de uma comunicação *top-down* entre a camada de tomada de decisão e a camada de vigilância. Exemplificando, temos um cenário em que um Agente de Tomada de Decisão, após receber uma mensagem ou um conjunto de dados dos Agentes de Vigilância, poderá no momento em que desempenhar uma análise, necessitar de mais informações. Neste ponto, novos Agentes de Vigilância deverão ser acionados e mais informações deverão ser coletadas, na tentativa de se conseguir uma decisão com um maior grau de certeza.

A ampliação do SDI para atender uma nova configuração de ataque poderá envolver o desenvolvimento e conseqüente adição de novos agentes em uma única camada ou ainda, a criação de um novo cenário que envolverá a adição de agentes em todas as camadas. A seguir, serão discutidas, com maiores detalhes, as funções de cada uma destas camadas.

4.3.1 A Camada 1 – Agentes de Vigilância

Esta camada, representa o primeiro nível de agentes do sistema proposto. Este conjunto de agentes será responsável pela monitoria, coleta de informações, testes de ambiente e ajustes de configuração a partir de arquivos de perfil de segurança que serão alocados estrategicamente no ambiente ao qual se quer proteger.

Uma analogia para os Agentes de Vigilância seria compará-los a guardas noturnos em uma empresa. Estes, ao invés de serem alocados estaticamente em pontos estratégicos do ambiente, seriam responsáveis em fazer a “ronda” pelo sistema em busca de portas abertas ou padrões que caracterizem possíveis intrusões. A seguir, são relacionados um conjunto de agentes de vigilância e suas respectivas funções:

- **Agentes de verificação de usuários conectados ao sistema:** Agente responsável pela obtenção de uma lista contendo os usuários conectados a um servidor, a hora de início e a origem destas conexões;

- **Agente de verificação de hospedeiros:** agente responsável pela verificação dos hospedeiros em execução no ambiente;

- **Agente de captura de pacotes:** agente que trabalha em modo promíscuo na rede ativando a captura de pacotes (*sniffing*);

- **Agentes de *scanning*:** conjunto de agentes que, através de tentativas de quebra de segurança, irão validar os serviços, processos e demais atividades que vão contra a política de segurança do ambiente;

- **Agente de verificação de serviços:** agente responsável pela verificação dos serviços em execução em máquinas servidoras;

- **Agente de monitoria dos serviços:** agente responsável por monitorar e filtrar as requisições de entrada dos serviços da rede, fornecendo informações para agentes da camada superior;

- **Agente de verificação de utilização de recursos do sistema:** agente responsável pela monitoria da utilização de recursos de sistema como CPU e *Hard Disk*, em momentos considerados estratégicos e não-críticos;

- **Agente especializado em identificação de *backdoors*:** agente que, com base nos padrões conhecidos e divulgados, irá rastrear o sistema em busca de *backdoors*. Este agente deverá apresentar uma grande facilidade em sua capacidade de reconfiguração com base em novos conhecimentos adquiridos, tendo em vista que constantemente são relatadas novas formas desse tipo de ataque;

- **Agente de validação de serviços:** agente responsável pela validação dos serviços em execução. Cabe a este agente a configuração do *profile* que autoriza a execução de um ou mais serviços em uma máquina/servidor do ambiente;

- **Agentes de configuração de *Profiles*:** com base na percepção de novas formas de intrusão ou variação do perfil de utilização dos usuários, proceder com as devidas (re)configurações dos *profiles*. Esses arquivos irão conter informações de padrões de normalidade de utilização do sistema, como: padrão dos horários de conexão de usuários; estatísticas de utilização de dispositivos do sistema; etc.

4.3.2 A camada 2 – Agentes de Tomada de Decisão

Nesta camada, encontram-se os agentes que exercem todas as funções de tomada de decisão no sistema, constituindo-se o "cérebro" do mesmo. Um agente desta camada irá receber uma mensagem ou um conjunto de dados dos agentes da camada inferior (a camada de Vigilância) e, com base em uma análise criteriosa destas informações, poderá identificar uma intrusão (ou tentativa), no momento de sua ocorrência, ou ainda, acionar novos Agentes de Vigilância para a coleta de informações complementares.

Em ações mais simples, estes agentes podem identificar uma anomalia, ou uso indevido, simplesmente comparando os dados obtidos com padrões de utilização do sistema (perfis de utilização). Entretanto, como esta camada representa o ponto de inteligência do sistema, deverão ser implementados agentes dotados de características de inteligência artificial para se alcançar um bom nível de reconhecimento de ações indevidas.

Entre essas características, é desejável que estes agentes sejam capazes de aprender coisas novas e se adaptarem a novas situações, em vez de simplesmente fazerem o que lhes foi atribuído. Entre os estímulos de ativação da aprendizagem, deve-se utilizar as ações tomadas pelo administrador ao ser notificado pelos agentes da camada de nível superior (a camada de notificação). Dessa forma, fica evidente a necessidade do desenvolvimento de agentes que exerçam funções de sistemas especialistas, o que poderá ser feito com a utilização das modernas técnicas de redes neurais e algoritmos genéticos, comumente referenciados em trabalhos relacionados à Inteligência Artificial.

A seguir são apresentados um conjunto de agentes de tomada de decisão e suas respectivas funções:

- **Agente de controle:** agente responsável pelo acionamento dos demais agentes. Constituindo-se o principal agente desta camada, este ficará em execução contínua à espera de uma mensagem de um agente da camada de vigilância. Ao receber e identificar a mensagem, este agente irá consequentemente acionar (ativar) o agente especializado em atender àquela chamada;

- **Agente de verificação de usuários anômalos:** agente que, baseado em um padrão de normalidade (*perfis* de usuário, como: horário de utilização, principais serviços utilizados, utilização de recursos do sistema, origem da conexão, etc), procura desvios do comportamento padrão, utilizando modelos estatísticos ou sistemas especialistas. Dessa forma, também será capaz de determinar possíveis personificações de usuários, baseando-se em uma análise do comportamento passado e na determinação de comportamentos que fujam ao padrão esperado de um comportamento aceitável;

- **Agente de verificação de hospedeiros não autorizados:** agente responsável pela verificação de existência de hospedeiros não autorizados na rede. Cabe a este agente a responsabilidade de validação dos ambientes servidores de agentes inseridos no sistema;

- **Agente de Análise de pacotes:** agente que recebe os pacotes do agente de captura (Camada de Vigilância) e procede com sua análise. As atividades desempenhadas por este agente representam um nível de complexidade elevado, uma vez que será dotado de algum mecanismo de tomada de decisão. Uma proposta inicial seria a conversão do módulo de análise do ambiente ACME! [CANSIAN, 1997] para o desempenho desta função;

- **Agente de identificação de pontos falhos:** agente que irá identificar os pontos falhos no ambiente com base nas informações dos agentes de *scanning;*

- **Agente de verificação de serviços:** agente responsável pela verificação da integridade e autorização dos serviços em execução em máquinas servidoras;

- **Agente de monitoria de *logs*:** agente que ativa a captura de informações de *logging* adicional quando um nível de periculosidade mais elevado é notificado;

- **Agente de monitoria de usuário indevido:** com base na detecção de uma possível personificação, proceder com o acompanhamento deste usuário em busca de assinaturas de ataque;

- **Agente de identificação de novos padrões de ataque:** agentes inteligentes responsáveis pelo aprendizado e identificação de novos padrões de ataque.

4.3.3 A camada 3 – Agentes de Notificação

Esta é a camada menos populosa em quantidade de agentes. Os agentes desta camada são responsáveis por, com base nas mensagens recebidas da camada 2 (Agentes de Tomada de Decisão), notificar o administrador da rede e acionar os agentes da camada 4 (Agentes de Reação). Dessa forma, toda vez que os Agentes de Tomada de Decisão identificarem um nível de periculosidade acima do aceitável ou a necessidade de atualização de algum novo padrão identificado, Agentes de Notificação entrarão em ação.

A princípio, pode-se pensar que os agentes desta camada desempenham funções muito elementares, o que justificaria a agregação de suas funções à camada 4(ocasionando a eliminação da camada 3). Entretanto, uma decisão tomada na camada de nível 2 poderá necessitar de diversas formas de notificação, ocasionando a construção de um agente muito complexo, quer seja agregando suas funções a esta camada ou em uma camada de nível superior do modelo. Isso iria contra a proposta de pequenos agentes desempenhando funções específicas na tentativa de minimizar a degradação do ambiente.

- **Agente de notificação de usuário indevido:** agente que irá alarmar a suspeita de uma personificação utilizando o sistema;

- **Agente de notificação de hospedeiro não autorizado:** agente que irá notificar a existência de hospedeiros não autorizados no ambiente;

- **Agente de notificação de tentativas de ataque:** agente que irá notificar as tentativas de conexão não autorizadas no ambiente;

- **Agente de notificação de serviço não autorizado:** agente que irá notificar a execução de algum serviço não-autorizado na rede;

- **Agente de notificação de furos de segurança:** Notificar furos de segurança provenientes de testes de *scanning;*

- **Agente de notificação de utilização de recursos de sistema:** agente que irá notificar a suspeita de utilização indevida de recursos do sistema;

- **Agente de notificação de tentativas de intrusão:** agente que irá notificar tentativas de intrusão por utilização de técnicas como quebra de senha por força bruta;

- **Agentes de acionamento de contra ataque:** Conjunto de agentes responsáveis por acionar cada ação de reação e contra-ataque;

- **Agente de notificação de identificação de novos padrões:** Agente que irá notificar a necessidade de reconfiguração ou desenvolvimento de novos agentes para atender um novo padrão de intrusão identificado.

4.3.4 A camada 4 – Agentes de Reação

Esta camada apresenta um conjunto de agentes que serão acionados pelos agentes da camada 3 (Agentes de Notificação) ou ainda, pela ação direta do administrador humano. Responsáveis por reagir (contra atacar), recuperar e reconfigurar o sistema, estes agentes representam a última instância de recursos do sistema modelado.

- **Agentes de encerramento de conexão:** agente responsável por cancelar e bloquear uma conexão para um possível usuário anômalo;

- **Agentes de exclusão de hospedeiros:** agente responsável pela remoção de hospedeiros não autorizados no ambiente;

- **Agente de corte de conexão:** agente responsável pelo acionamento do corte e/ou bloqueio da conexão de um possível intruso;

- **Agentes de eliminação de furos de segurança:** conjunto de agentes responsáveis pela reconfiguração dos serviços na tentativa de eliminação de furos de segurança;

- **Agente de bloqueio de serviço:** agente responsável pelo bloqueio da execução de serviços não autorizados;

- **Agente de recuperação de arquivos alterados:** agente que cuida da recuperação do estado inicial, baseado em um repositório, caso seja efetuada uma alteração não validada em um arquivo,

- **Agentes de reconfiguração:** agente responsável pela recuperação do estado anterior de serviços autorizados;

- **Agente de ativação de dispositivos auxiliares:** agente responsável pela ativação de dispositivos auxiliares à captura de informações sobre o usuário interno. Um exemplo de um dispositivo auxiliar poderia ser uma câmera de vídeo.

4.4 Classificação do Sistema

Conforme visto no capítulo 2, as principais metodologias de classificação para os Sistemas de Detecção de Intrusão são expressas em termos da forma como o sistema aborda o problema de detectar a intrusão e em termos do tratamento dos dados.

Uma vez que a detecção de anomalia identifica atividades intrusivas como sendo um sub-conjunto que foge da atividade normal, o sistema proposto possui um conjunto de agentes que procuram quantificar o comportamento usual ou aceitável, armazená-lo em *profiles* de usuário e posteriormente, identificar outros comportamentos irregulares como intrusivos. Entretanto, o sistema também possui agentes que procuram por ataques que podem ser precisamente identificados pela forma como acontecem, ou seja, intrusões que seguem um padrão bem definido de ataque (assinaturas de ataque), características estas do modelo de detecção de uso indevido (abuso).

Assim, em relação à forma com que aborda o problema de detectar a intrusão, o sistema proposto apresenta-se como um híbrido entre o modelo de detecção de anomalia e o modelo de

detecção baseado em uso indevido. Isso também pode ser considerado uma vantagem significativa deste sistema, uma vez que sistemas híbridos monolíticos apresentam-se como sistemas complexos que implicam em severas penalidades de performance no ambiente a ser monitorado, o que não ocorre com a proposta modular.

Em termos de tratamento dos dados, o sistema apresenta-se como um híbrido entre um modelo baseado em *host (host based)* e um modelo baseado em redes *(network based).* Entre as características de um sistema baseado em *host*, o sistema apresenta um conjunto de agentes que, baseados em perfis de utilização de um equipamento, procuram por desvios de comportamento padrão, utilizando-se modelos estatísticos ou sistemas especialistas. Entretanto, também possui agentes que monitoram o tráfego da rede, capturam pacotes e procuram por "impressões digitais" do ataque, acontecendo em tempo real.

4.5 Cenários de execução do ambiente modelados com DFD

Neste ponto, torna-se interessante a representação de alguns cenários de execução do sistema, visando sua melhor compreensão e uma perfeita visualização do relacionamento entre as camadas do modelo apresentado. Para essa representação, foi utilizado uma ferramenta lógica conhecida como Diagrama de Fluxo de Dados (*DFD),* por ser uma ferramenta simples, de fácil entendimento e largamente utilizada na modelagem de sistemas de informação. Como os símbolos utilizados no DFD não são físicos, ele mostra a essência da lógica subjacente do sistema a ser modelado.

A figura 4-2 apresenta os símbolos utilizados e sua representação neste contexto. Maiores informações sobre DFDs podem ser observadas em [GANE, 1988] e [GANE & SARSON, 1994].

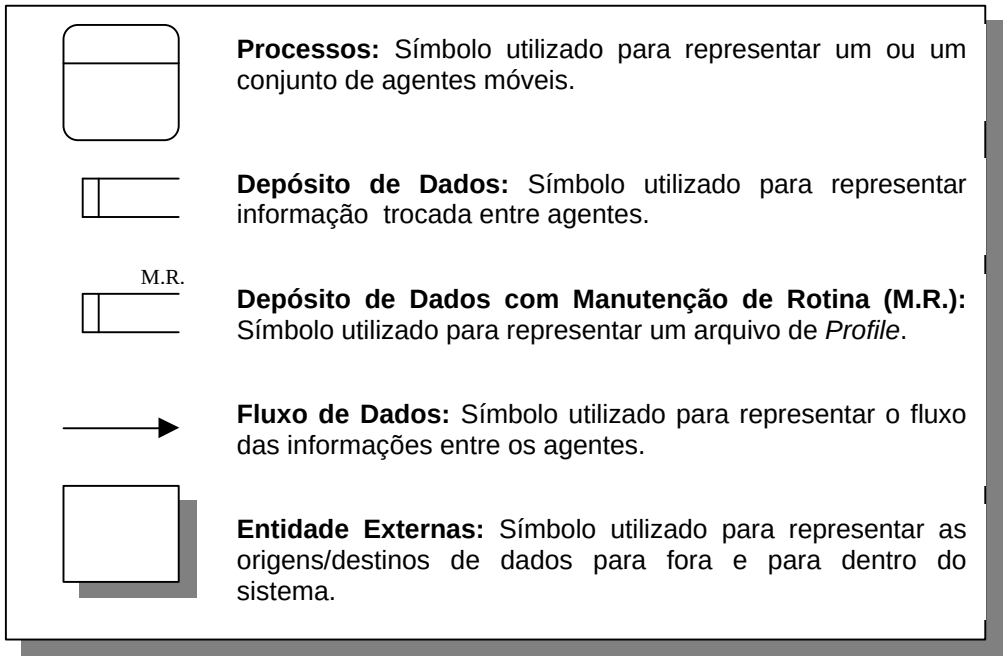

Figura 4-2- Representação dos elementos de DFD utilizados na modelagem

Nas próximas seções, serão apresentados os modelos (utilizando-se DFDs), para alguns cenários de execução do sistema. Cada cenário está representado por um modelo de alto nível composto por 4 processos que estão associados respectivamente a cada uma das camada do modelo apresentado na figura 4-1. Assim, a numeração utilizada em cada processo, além de identificar a seqüência lógica de execução do cenário, faz a associação do processo à sua respectiva camada no modelo. Como exemplo, temos que o processo 1 contém um ou mais agentes da camada 1 (a Camada de Vigilância) e assim sucessivamente.

Dependendo da complexidade, um processo(agente) neste modelo de alto nível pode ser "expandido" para tornar-se um novo diagrama (conjunto de agentes). Cada cenário representa uma fronteira de automação para o desempenho de uma função de segurança computacional no sistema proposto. Assim, a criação de um novo cenário de execução corresponderá à implementação e posterior inserção de novas funções no sistema proposto.

4.5.1 Cenário 1: Identificação de Usuários Anômalos

Um intruso (interno ou externo) que consiga um *username* e uma senha válidos, poderá abrir uma conexão e ter acesso ao sistema. Uma vez dentro do sistema (mesmo em acesso

comum, ou seja, não como *root*) poderá usar diversas técnicas disponíveis para executar suas intenções, conseguir privilégios de *root* ou ainda, agir ilegalmente em nome do usuário legítimo.

Na tentativa de identificação deste tipo de ataque, este cenário apresenta as características de um Sistema de Detecção de Intrusão baseado em anomalias. O objetivo é analisar comportamentos que possam identificar conexões que fujam dos padrões previamente estabelecidos para cada usuário. Para isso, o sistema possui: agentes que coletam uma lista contendo informações dos usuários conectados ao sistema (horário de *login*, origem da conexão, etc); agentes que irão proceder com a identificação de possíveis anomalias tomando como base *profiles* previamente estabelecidos para os usuários; agentes de notificação do grau de suspeita e ainda, agentes para execução de medidas reativas, tais como bloqueio de conexão.

O fluxo lógico da ação dos agentes descritos anteriormente é modelado e apresentado na figura 4-3. Conforme pode ser observado, apesar de todos os processos representarem agentes autônomos que estarão em ação sem a supervisão humana, o administrador poderá interagir diretamente com cada um deles, utilizando-se das primitivas de controle de agentes. Entre as primitivas encontram-se: criação, clonagem, envio, recuperação, suspensão de execução, interrupção de processamento, etc. Isso também é válido para os demais cenários apresentados.

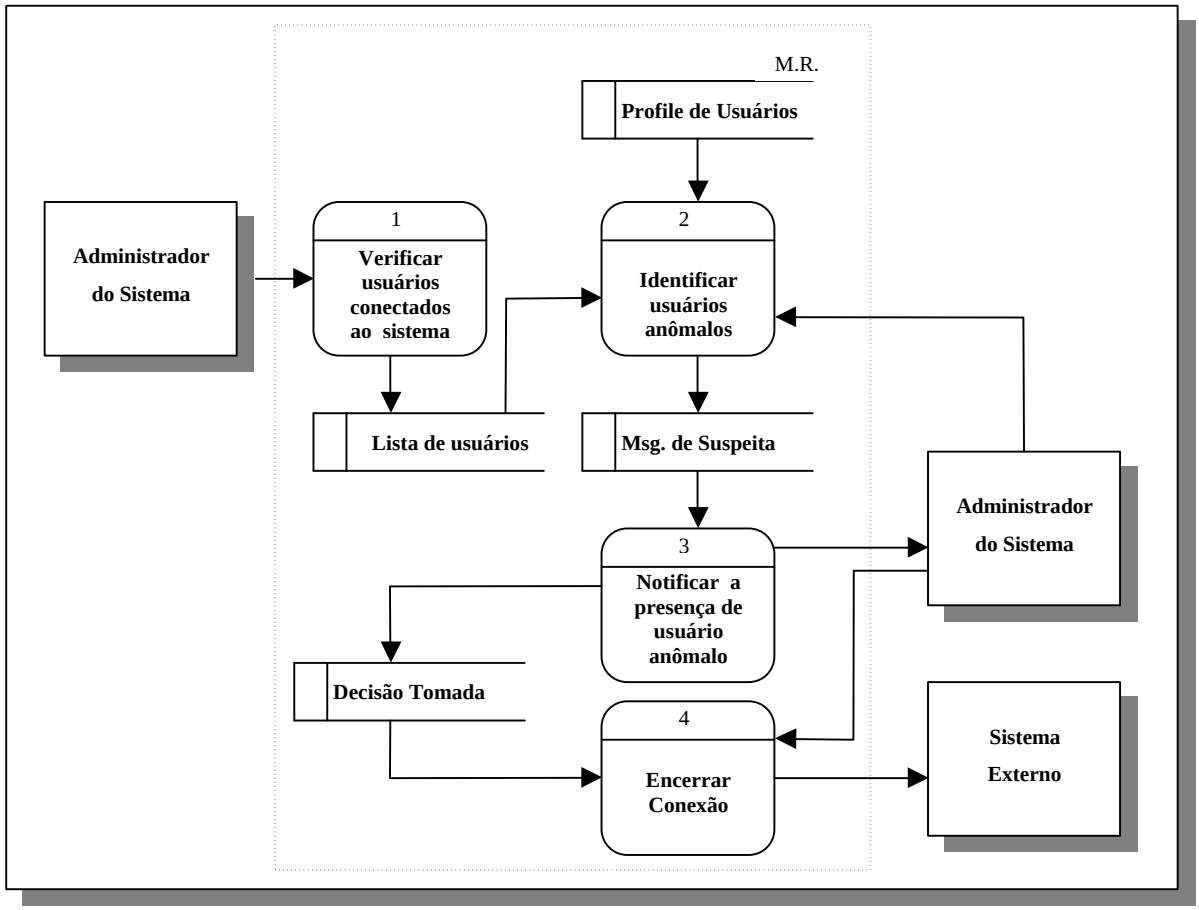

Figura 4-3 - Modelagem do Cenário de Identificação de Usuários Anômalos

4.5.2 Cenário 2: Identificação de Hospedeiros Não Autorizados

Conforme apresentado anteriormente, agentes móveis necessitam de um ambiente de hospedagem para que possam viajar pela rede desempenhando suas funções. Desta forma, torna-se necessário um processo contínuo de validação dos hospedeiros existentes no sistema. Isso reflete-se na tentativa de não permitir a execução de hospedeiros *hostis*, uma vez poderiam mascarar informações para os agentes de segurança ou ainda, prover uma porta de intrusão para agentes *hostis*.

A figura 4-4 apresenta o modelo para as funções dos agentes envolvidos neste cenário

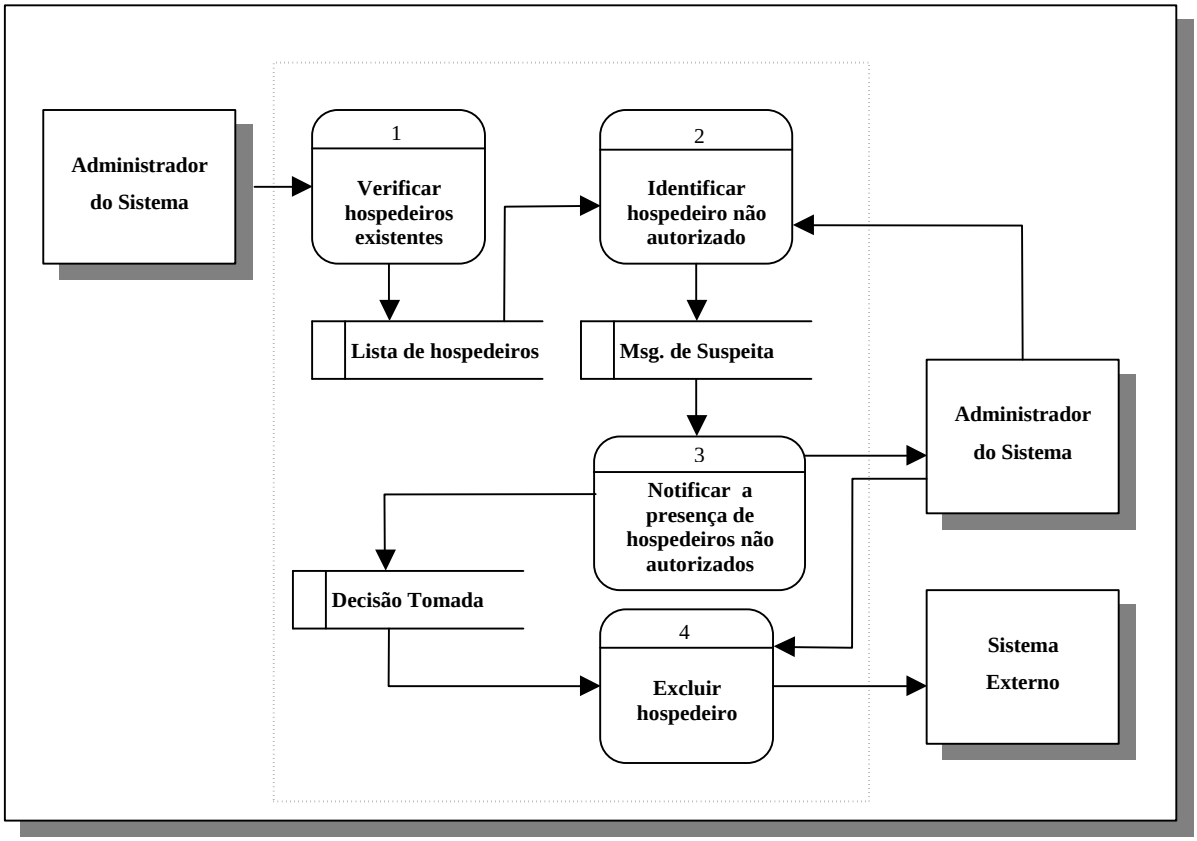

Figura 4-4 - Modelagem do Cenário de Identificação de Hospedeiros Não Autorizados

4.5.3 Cenário 3: Identificação de Tentativas de Ataque

Além de identificar e reagir as intrusões no momento em que elas acontecem, o sistema também apresentará características de identificação de tentativas de ataque. Estas informações serão úteis para se quantificar essas tentativas e auxiliar na elaboração da política de segurança da organização.

Um tipo de ataque simples e comum é o denominado *doorknob*. Ele consiste basicamente em abrir uma conexão *telnet* ou ftp em alguma máquina e, pelo método de tentativa e erro, conseguir encontrar uma senha válida para consequentemente obter acesso ao sistema. Apesar de parecer um método bastante ineficaz, ainda existem redes mal configuradas com usuários e até

mesmo administradores negligentes que freqüentemente possuem senhas simples. Exemplos de senhas simples possíveis para a conta *root* seriam: *root*, *toor*, *administrator*, *operator*, *rotarepo*, *secret*, o nome do administrador, e diversos outras senhas comuns. Existem ainda outras contas no sistema que podem ser exploradas como: *access, admin, archie, audit, backup, bulletin, demo, bbs, bin, diagnostic, ftp, games, gopher, help, guest, irc, lib, local, manager, news*, etc.

No exemplo da figura 4-5, o intruso atacando o servidor de ftp Vitima tenta três senhas para a conta *root*: *root*, *toor* e *administrator*.

```
220 Vitima FTP server (Version 6.20 Tue Jun 27 14:38:19 GMT-0300  1995)
   ready.    If you have problems contact <operator@vitima.com.br>.  .
   331 Password required for root..
   PASS root.
   530 Login incorrect..
   USER root.
   331 Password required for root..
   PASS toor.
   530 Login incorrect..
   USER root.
   331 Password required for root..
   PASS administrator.
   530 Login incorrect..
```

Figura 4-5 – Exemplo de ataque *doorknob*

A figura 4-6 apresenta o fluxo de informação entre o os agentes que irão formar o cenário que irá fornecer a capacidade de identificação de tentativas de ataque ao sistema proposto e consequentemente prover características de um sistema de detecção de intrusão baseado em rede (*network based).*

Este cenário é composto por: um agente que irá trabalhar em “modo promíscuo” capturando pacotes que trafegam pela rede; um agente dotado de funções de um sistema especialista que tomará decisões de quais conexões terão seu conteúdo monitorados; um agente de notificação do grau de suspeita alcançado e um agente de tomada de decisão. Uma decisão tomada poderá ser a proibição de conexão para um determinado usuário.

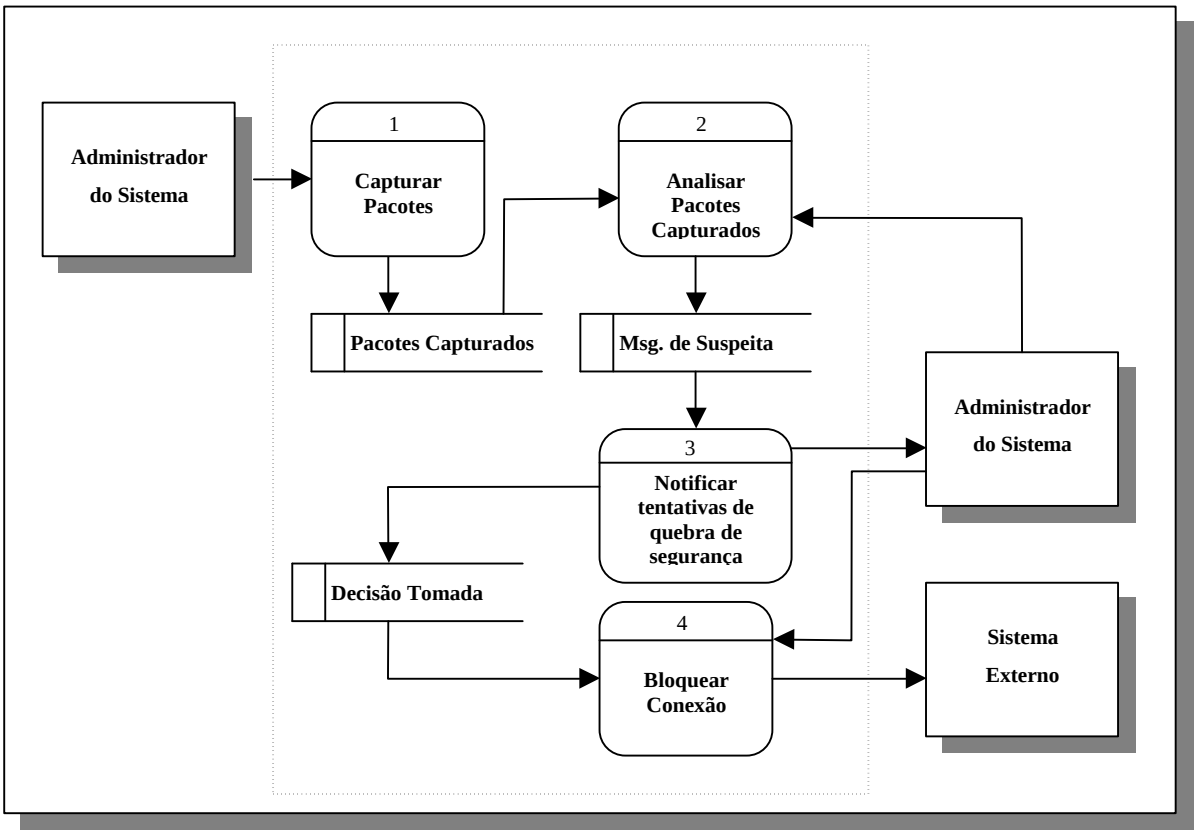

Figura 4-6 - Modelagem do Cenário de Identificação de Tentativas de Ataque

4.5.4 Cenário 4: Realização de *Scanning*

Este cenário apresenta um conjunto de agentes que irão desempenhar tentativas de quebra de segurança para conseqüente identificação "furos" no ambiente e validação da política de segurança adotada.

Observando o modelo para este cenário, apresentado na figura 4-6, verifica-se a necessidade de expansão do processo 1 (Realizar *scanning*). Após a expansão, o cenário será composto por um conjunto de agentes especializados nas mais diversas formas de *scanning:* quebra de senha por força bruta; *Portscanning*, etc. Como este processo está associado à camada de Verificação, todos os processos derivados irão realizar funções da mesma camada.

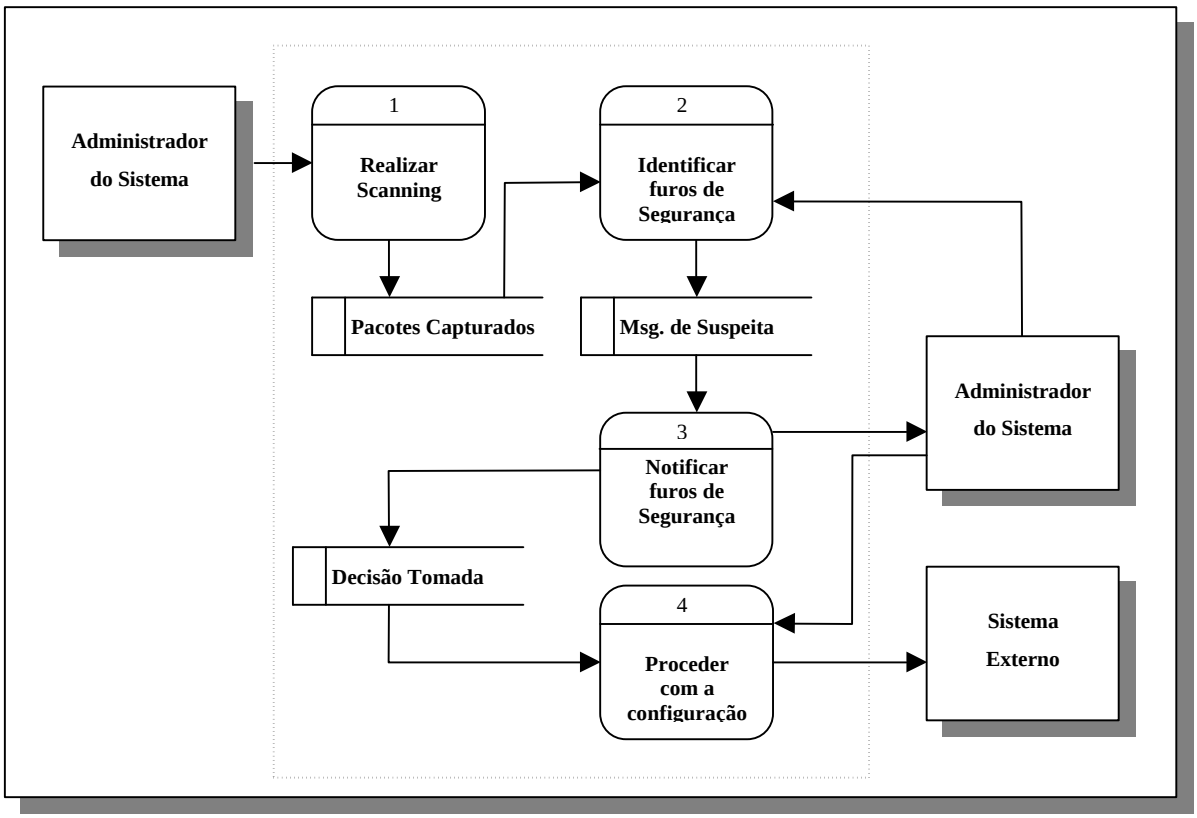

Figura 4-7 - Modelagem do Cenário de Realização de *Scanning*

4.5.5 Cenário 5: Identificação de Execução de Serviços Não Autorizados

O conjunto de agentes apresentados neste cenário cuidará da identificação de serviços não autorizados em execução no ambiente protegido.

A partir de sua identificação, agentes de notificação cuidarão de informar o administrador, invocar um agente de reação cuja função será de bloquear o serviço ou ainda, trocar informações com o cenário de identificação de usuários anômalos para que este possa proceder com a identificação de possíveis suspeitos.

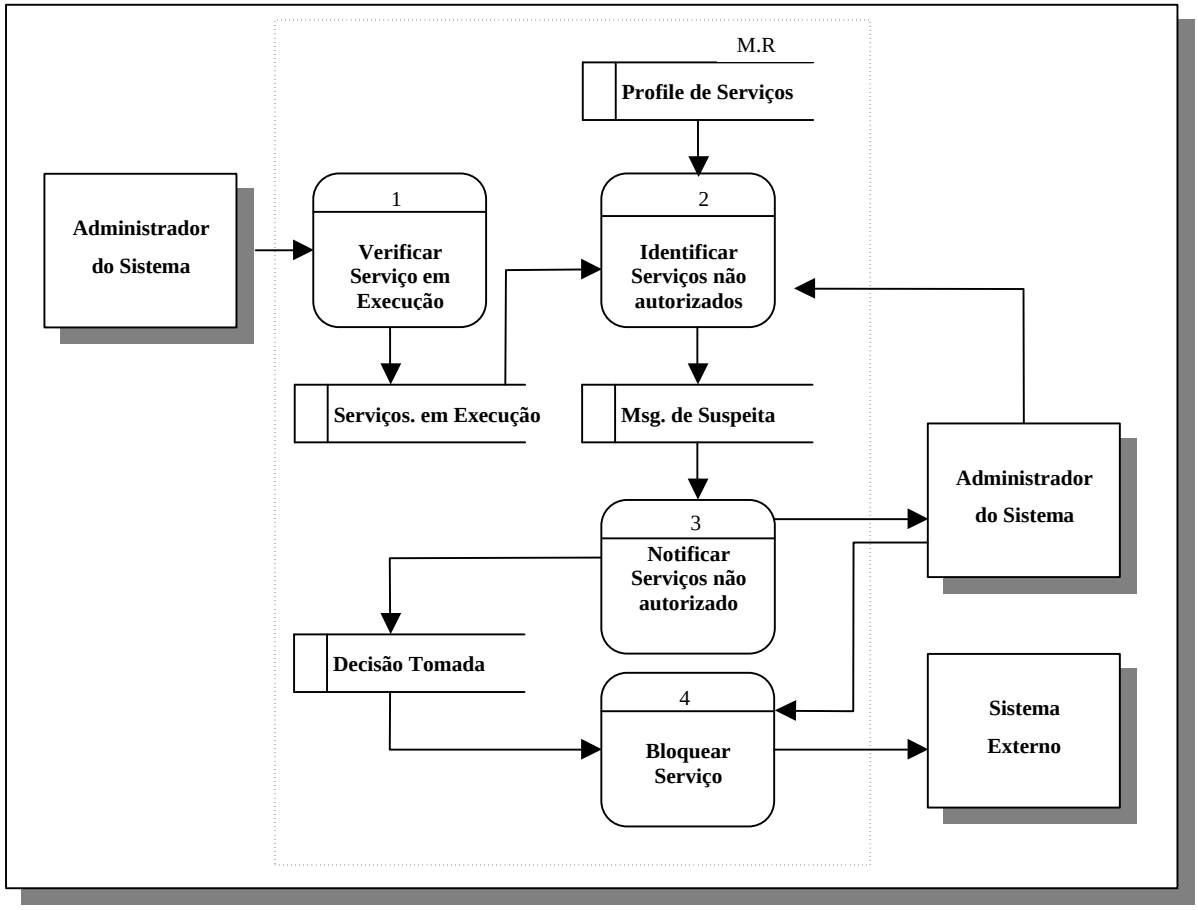

Figura 4-8 - Modelagem do Cenário de Identificação de Execução de Serviços Não Autorizados

4.5.6 Cenário 6: Identificação de *Backdoors*

A utilização de *backdoors* como métodos de proferir intrusão está cada vez mais acentuada. A cada dia são relatados novos "furos" nos mais diversos programas, como por exemplo o recente relato da *General Tech Informática* sobre a existência de um novo *backdoor* que se instala utilizando as macros existentes no Word 97. *Backdoors* como o *NetBus*, *Back Orifice* e o *Wincrash* são softwares muito fáceis de serem utilizados e encontrados facilmente na Internet para *download* e utilização.

Estes, quando instalados, deixam "assinaturas" que podem identificar sua utilização. Assim, o cenário modelado na figura 4-8 apresenta um conjunto de agentes para monitoramento, identificação, notificação e eliminação de *backdoors*.

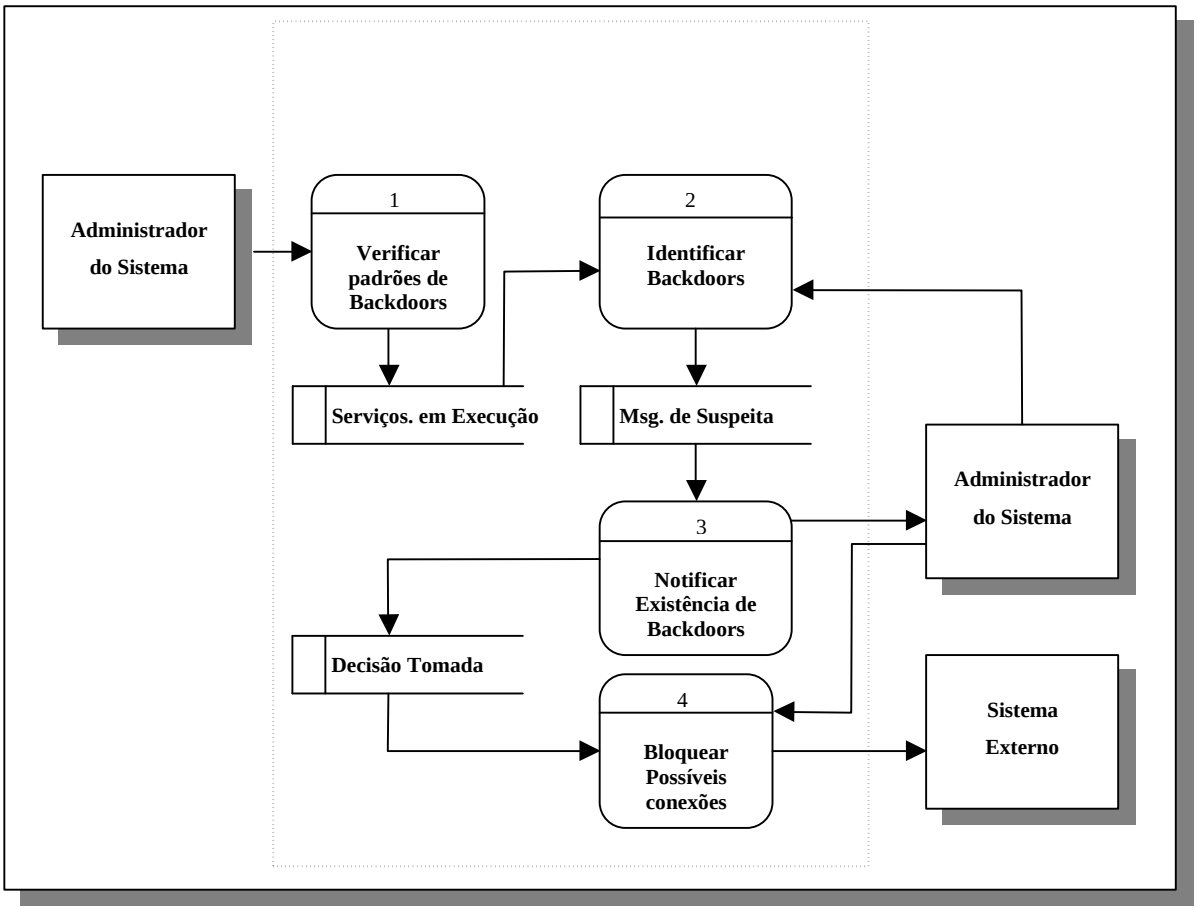

Figura 4-9 - Modelagem do Cenário de Identificação *Backdoors*

4.6 Características desejáveis do Ambiente

No desenvolvimento de um ambiente de detecção de intrusão conforme o que discutimos até o presente momento, algumas características tornam-se desejáveis e desta forma, o sistema deve:

- estar em execução contínua com o mínimo possível de supervisão humana;
- ser capaz de executar em *background* no ambiente que está sendo observado e relatar suas conclusões;

- impor o mínimo de *overhead* possível no ambiente onde está sendo executado, de forma a não interferir nas operações normais;

- apresentar facilidades de configuração para atender a política de segurança do ambiente que está monitorando;

- adaptar-se às mudanças do ambiente e dos comportamentos dos usuários através do tempo (permissão para execução de novas aplicações, usuários trocando de atividade ou novos recursos sendo utilizados);

- monitorar um grande número de *hosts* enquando providencia resultados em tempo hábil e de maneira exata;

- apresentar uma moderada degradação de serviço, uma vez que algum componente do sistema pode ter seus serviços interrompidos por qualquer razão, o resto deverá ser afetado da menor forma possível;

- permitir reconfiguração dinâmica, uma vez que um grande número de *hosts* estão sendo monitorados, torna-se impraticável reinicializar o IDS toda vez que uma novo cenário for implementado.

4.7 Trabalhos existentes

Recentemente, o COAST *(Computer Operations, Audit and Security Technology, Computer Science Departament at Purdue University*) liberou uma implementação de um ambiente que segue as idéias do sistema proposto por Crosbie e Spafford [CROSBIE & SPAFFORD, 1995a, 1995b]. Este ambiente denominado *Autonomous Agents for Intrusion Detection* (AAFID) foi implementado utilizando-se a linguagem de *script*s 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 & 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, detecção baseada em uso indevido e a criação de objetos específicos para tratar os diversos tipos de ataque.

Em sua tese de mestrado, [REAMI, 1998] apresenta a especificação e o protótipo de um ambiente de gerenciamento de segurança computacional apoiado por agentes móveis. O ambiente especificado e prototipado pode ser considerado um avanço importante, pois, além de fornecer recursos tecnológicos avançados e consoantes com o desenvolvimento da área, permite uma abordagem holística do problema do gerenciamento de segurança.

4.8 Considerações Finais

Neste capítulo é apresentado uma modelagem para um sistema de detecção de intrusão baseado em agentes móveis e a definição de alguns cenários de sua utilização. Este sistema tem por finalidade a minimização dos custos apresentados por um SDI monolítico. Ele utilizará uma grande quantidade de pequenos agentes móveis para desempenharem todas as ações de monitoria, tomada de decisão, notificação e reação a tentativas de intrusão. Cada agente opera independentemente um dos outros, porém, todos cooperam na monitoria do sistema formando um complexo SDI. Como visto, esta abordagem apresenta vantagens significativas em termos de *overhead*, escalabilidade e flexibilidade.

5 Tecnologias de Suporte ao Desenvolvimento do Sistema Proposto

5.1 Considerações Iniciais

Diversos sistemas de agentes móveis e *frameworks* de suporte ao desenvolvimento de aplicações que utilizam esta tecnologia vem sendo propostos e implementados.

Este capítulo apresenta uma discussão sobre as principais tecnologias de suporte ao desenvolvimento de sistemas baseados em agentes móveis, dando ênfase ao *Aglets Software Development Kit (ASDK),* uma API Java Aglet (J-AAPI) desenvolvida pela *IBM Corporation.* Essa foi a plataforma utilizada no desenvolvimento dos cenários iniciais do sistema proposto.

5.2 Visão geral das Tecnologias de Agentes Móveis

Tecnologias de agentes móveis incluem linguagens de programação e seus correspondentes suportes à execução. Fuggeta, Picco & Vigna [FUGGETA et all, 1998] apresentam alguns mecanismos existentes de suporte a Sistemas de Código Móvel *(Mobile Code Systems - MCS)* e suas principais características. Segundo eles, os componentes de *um Ambiente Computacional* para código móvel *(Computational Environments - CE)* podem ser diferenciados em *Unidades em Execução* (*Executing units – EU)* e *Recursos*. Exemplos típicos para *Unidades em Execução* são processos *single-threaded* ou *threads* individuais de um processo *multi-threaded. Recursos* representam entidades que podem ser compartilhada entre múltiplas *Unidades em Execução*, como um arquivo em um sistema de arquivo.

Existem duas formas de mobilidade dos sistemas de código móvel [FUGGETA et all, 1998]:

- Mobilidade forte: é a habilidade de um *Sistemas de Código Móvel* (chamado MCS forte) permitir a migração do código e o do estado de execução de uma *Unidade em Execução (EU)* para diferentes *Ambientes Computacionais (CEs).*

- Mobilidade fraca: é a habilidade de um *Sistemas de Código Móvel* (chamado MCS fraco) permitir a transferência de código através de diferentes *Ambientes Computacionais* (*CEs*); o código pode ser acompanhado por alguns dados de inicialização, mas nenhuma migração de estado de execução é envolvida.

5.2.1 Agentes Móveis com Java

Desenvolvido pela Sun Microsystems, Java tem chamado a maioria das atenções e expectativas em mobilidade de código. A meta original dos projetistas da linguagem foi fornecer portabilidade, clareza, fácil aprendizado e uma linguagem orientada a objetos que contribuísse para o crescimento da Internet. O compilador Java traduz programas fontes Java em um código intermediário e independente da plataforma chamado *Java Byte Code*, que é interpretado por uma *Java Virtual Machine* (JVM).

A seguir, são apresentadas as principais vantagens que a linguagem Java introduz no desenvolvimento de sistemas baseados em agentes móveis:

1) Independência de plataforma: Java foi desenvolvida para operar em ambientes heterogêneos, possibilitando que uma aplicação execute em qualquer computador em uma rede. Para isso, o compilador gera *byte codes* independentes da arquitetura. Para esse código ser executado, o sistema Java *runtime* deve estar presente. A linguagem Java não tem nenhum aspecto dependente de plataforma. Tipos de dados primitivos são rigorosamente especificados para não depender do processador ou do sistema operacional. Isso permite criar um agente móvel sem conhecimento das características dos computadores em que ele irá rodar.

2) Execução segura: Java foi elaborada para ser usada em ambientes distribuídos. Por causa disso, muita ênfase foi dada à segurança. Por exemplo: a) Java tem um modelo de ponteiros que elimina a possibilidade de sobrescrever a memória e corromper os dados;. b) Java simplesmente

não permite tipos de dados ilegais ou alguma aritmética de ponteiro ; c) Os programas não são capazes de obter acesso a dados privados em objetos para os quais eles não têm acesso, um arranjo que evita uma certa categoria de vírus; d) Se alguém modificar o *byte code*, o sistema *Java runtime* assegura que o código não será capaz de violar as semânticas básicas de Java; e) Java também tem um gerenciador de segurança para checar todas as operações potencialmente não seguras, assim como acesso a arquivos e conexões com a rede, para determinar se o programa tem permissão para executar essas operações. Consequentemente, a arquitetura de segurança de Java torna-a razoavelmente segura para hospedagem de agentes desconhecidos (*untrusted*).

3) Carregamento dinâmico de classes: Este mecanismo permite à máquina virtual carregar e definir classes em tempo de execução. Isso fornece um *name space* protegido para cada agente, permitindo executarem de forma segura e independente uns dos outros. O mecanismo de carregamento de classe é extensível e possibilita que as classes sejam carregadas via rede.

4) Programação *multithread*: Os agentes são por definição autônomos; um agente executa independentemente de outros agentes dentro do mesmo ambiente de execução. Permitir que cada agente execute em seu próprio processo (*thread*) é uma maneira de permitir que os agentes sejam autônomos. Java não somente permite programação *multithread* mas também suporta um conjunto de primitivas de sincronização. Essas primitivas permitem a interação de agentes.

5) Serialização de objetos: A característica chave de agentes móveis é que eles podem ser serializados e desserializados. Java fornece um mecanismo de serialização que pode representar o estado de um objeto em uma forma serializada suficientemente detalhada para o objeto ser reconstruído mais tarde. A forma serializada do objeto deve ser capaz de identificar a classe Java cujo estado do objeto foi salvo e restaurá-lo em uma nova instância.

Apesar do sistema da linguagem Java ser altamente satisfatória para criação de agentes móveis, nós devemos estar atentos a algumas negligências significantes. Algumas delas podem ser trabalhadas, outras são mais sérias e têm implicações para todo o projeto conceitual e desenvolvimento de sistemas de agentes móveis baseados em Java.

1) Suporte inadequado para controle de recursos: O sistema da linguagem Java não fornece meios para controlar os recursos consumidos pelos objetos Java. Por exemplo, um agente pode

iniciar um *loop* e desperdiçar ciclos do processador e também consumir recursos de memória. Esses dois exemplos relatados podem ocasionar um tipo específico de ataque de segurança denominado *denial of service*. Este é o tipo mais temido de ataque proveniente de agentes móveis, onde um conjunto de agentes migram para um computador e pegam todos os seus recursos, impossibilitando o seu controle pelo operador. Infelizmente, Java não fornece nenhuma maneira para o host limitar os recursos do processador e memória alocados pelo objeto ou *thread*. Desta forma, cláusulas especiais de validação de agentes ou até mesmo agentes de monitoria devem ser inseridas no ambiente hospedeiro a fim de evitar este tipo de problema.

2) Nenhuma proteção de referência: Os métodos públicos do objeto Java são disponíveis para outros objetos que têm uma referência a eles. Não há nenhuma maneira na qual o agente pode, diretamente, monitorar e controlar o acesso de outros agentes a seus métodos públicos.

Uma solução prática e poderosa para esse problema é inserir um objeto *proxy* entre o agente que faz a chamada e o que é chamado e, desta forma, controlar o acesso (figura 5-3).

3. Nenhum suporte para preservação e recuperação do estado de execução: Atualmente, é impossível em Java recuperar o estado de execução de um objeto. Informações como o *status* do *program counter* e *frame stack* é permanentemente território proibido para programas Java. Então, para um agente móvel completar sua execução em um *host* remoto, ele deve confiar em valores de atributos internos e eventos externos para direcioná-lo. Um autômato pode ser embutido para assegurar que as execuções serão corretamente interrompidas e retornadas à execução.

5.2.2 Sistemas de Agentes Móveis Baseados em Java

A linguagem Java viabilizou a concepção de diversos sistemas experimentais de agentes móveis. Numerosos sistemas estão atualmente em desenvolvimento e a maioria estão disponíveis via Web. Exemplificando alguns destes ambientes, pode-se citar:

- **Aglets -** Criada pela IBM Corporation, constitui-se de uma API que tenta espelhar o modelo de *Applets Java*. A proposta de seu desenvolvimento era dar mobilidade aos

Applets [LANGE & OSHIMA, 1998]. Visto ser este o ambiente de desenvolvimento para este projeto, ele será explorado com maiores detalhes na seção 5.3.

- **Odyssey** - A General Magic Inc. criou o primeiro sistema de agentes móveis comercial, chamado Telescript. Entretanto, o Telescript teve pouco sucesso pois era baseado em uma linguagem e arquitetura de rede proprietárias. A Popularidade da Internet motivou a General Magic a reimplementar um sistema de agentes móveis baseado em Java chamado *Odyssey*. Este sistema simplesmente mapeou os conceitos do Telescript em classes Java, permitindo aos desenvolvedores criar suas próprias aplicações [MAGIC, 1998].

- **Concórdia** - Concebido pela Mitsubishi constitui-se de um *framework* para o desenvolvimento e gerenciamento de aplicações de agentes móveis. O Concórdia compreende múltiplos componentes, todos escritos em Java, os quais são combinados para prover um ambiente para aplicações distribuídas. Este sistema é simples e requer somente uma implementação padrão da máquina virtual Java. Seu ambiente é composto de um servidor e um conjunto de agentes. O Concórdia provê mecanismos de segurança para execução segura de agentes, além de suportar mecanismos de *checkpoint* para tolerância a falhas [MITSUBISHI, 1997].

- **Voyager** - Criado pela ObjectSpace, consiste de uma plataforma ORB (*Object Request Broker*) implementado em Java e com suporte a agentes. Voyager implementa os mecanismos tradicionais de troca de mensagens, somados à capacidade de objetos moverem-se através da rede como agentes. Uma desvantagem de Voyager é não oferecer mecanismos de segurança contra agentes não autorizados [SPACE, 1998].

Esses sistemas de agentes móveis Java possuem características em comum:

- Uma versão padrão da Máquina Virtual Java (JVM) e mecanismos de serialização de objetos Java;

- Um ambiente servidor de agentes encontra-se presente em todos os sistemas;

- Mecanismos de transporte e suporte para interação (troca de mensagens).

5.2.3 Outras linguagens para Agentes Móveis

Embora a maioria dos sistemas de agentes móveis existentes são baseados em na linguagem Java, existem outras linguagens em uso. As linguagens mais significantes *são Tcl, Scheme e Python.*

- *Agent Tcl:* Desenvolvido no Dartmouth College's, *Agent Tcl* é um sistema de agentes móveis onde agentes podem ser escritos em Tcl com suporte para mobilidade forte. Ele tem uma grande quantidade de serviços de navegação e comunicação, mecanismos de segurança, depuração e ferramentas de localização de agentes. O principal componente de um Agente Tcl é um servidor que, executado em cada máquina, permite a execução completa do agente. Quando um agente deseja migrar para uma nova máquina, ele chama uma simples função, agent_jump, que automaticamente captura o estado completo do agente e envia as informações deste estado para o servidor na máquina de destino. O servidor de destino inicia a execução Tcl, lê o estado de informação em seu ambiente de execução e reinicializa o agente no ponto exato de onde ele foi interrompido.

- *Ara:* Desenvolvido pela University of Kaiserslautern, Ara também é um sistema que suporta mobilidade forte baseado em Tcl e propõe uma plataforma para a execução segura de agentes móveis em redes heterogeneas. Agentes *Ara* são gerenciados pelo núcleo de um sistema independente de linguagem e por um interpretador de linguagem que dá suporte às linguagens C, C++ e Tcl.

- *TACOMA:* Um projeto da University of Tromso e Cornell University. Utilizando a mobilidade fraca, TACOMA focaliza o suporte para sistema operacional e como agentes podem ser usados para resolver problemas tradicionalmente direcionados para este sistema operacional. O sistema TACOMA é baseado em UNIX e no TCP. O sistema suporta agentes escritos em C, Tcl/Tk, Perl, Python e Scheme (Elk). Este sistema é totalmente desenvolvido em C.

A descrição de outros sistemas como Facile, M0, Obliq e Sumatra pode ser encontrado em [FUGGETA et all, 1998].

Alguns projetos baseados em Tcl anteciparam o movimento para suporte a múltiplas linguagens, algo que foi reforçado pelo suporte fornecido pela linguagem Java.

5.3 Java Aglets

O *Aglet* é um agente móvel Java que suporta os conceitos de execução autônoma e roteamento dinâmico em seu itinerário. Conforme pode ser observado na figura 5-1, *Aglets* são hospedados por um servidor *Aglet* de forma similar à forma como os *applets* são hospedados por um *Web browser*. O servidor *Aglet* provê um ambiente para os *Aglets* executarem e uma *Java Virtual Machine (JVM)* que, juntamente com o gerenciador de segurança do *Aglet,* tornam esse servidor seguro para receber e hospedar os *Aglets.*

Desta forma, a origem da palavra *Aglet* é simples: ela deriva do termo "*lightweight agent*" da mesma forma que um *Applet* deriva de "*lightweight application*". O termo *Aglet* é uma combinação das palavras *Agent* e *Applet.*

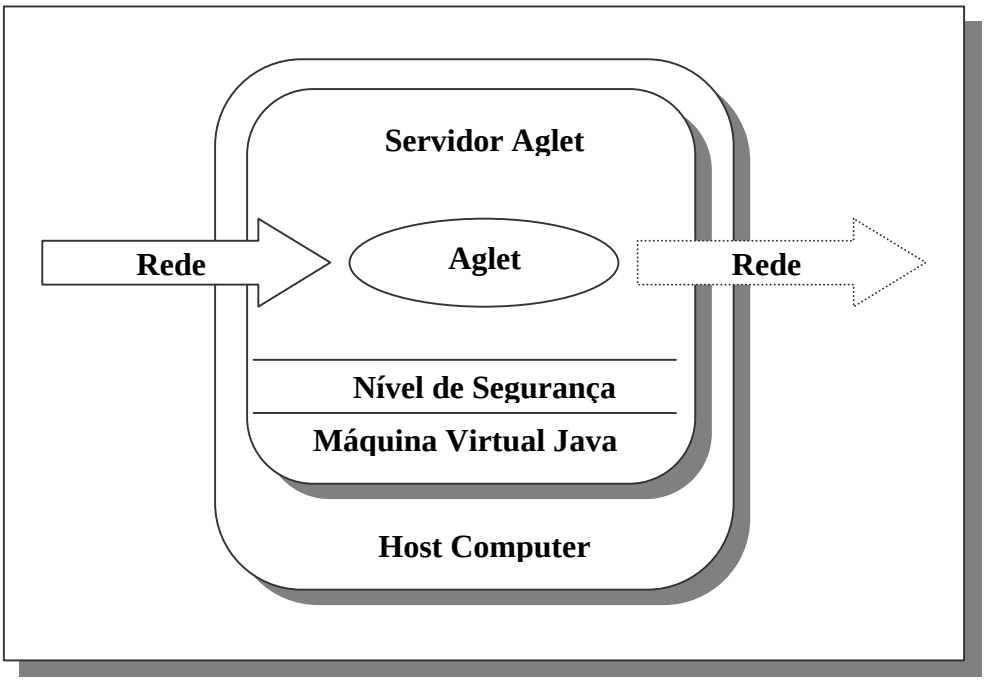

Figura 5-1 - Ambiente Seguro para *Aglets* visitantes [LANGE & OSHIMA, 1998]

A API *Aglet* é um kit de desenvolvimento de agentes ou , em outras palavras, um conjunto de classes Java e interfaces que permitem a criação de agentes móveis Java. O grupo *IBM Tokyo Research Laboratory* desenvolveu a API *Aglet* no Japão, tendo sua primeira versão liberada em 1995 em resposta à necessidade de uma plataforma uniforme para agentes móveis em ambientes heterogêneos como a Internet

Sendo a API *Aglet* desenvolvida em Java, um agente móvel desenvolvido com sua utilização será capaz de executar em qualquer máquina que suporta Java. Não é preciso informações sobre o hardware ou sobre o sistema operacional e dessa forma, a API *Aglet* espelha o modelo de *applet* em Java.

Uma implementação da API *Aglet*, denominada ASDK, pode ser encontrada para *download* no Web site da IBM Tokyo Research Laboratory's (http://www.trl.ibm.co.jp/aglets). O ASDK inclui o pacote API *Aglet*, documentação, exemplos de *Aglets* e o servidor de *Aglets* denominado Tahiti, apresentado na figura 5-2. O Tahiti é uma aplicação Java que permite ao usuário receber, gerenciar, e enviar *Aglets* para outros computadores que estão executando o Tahiti.

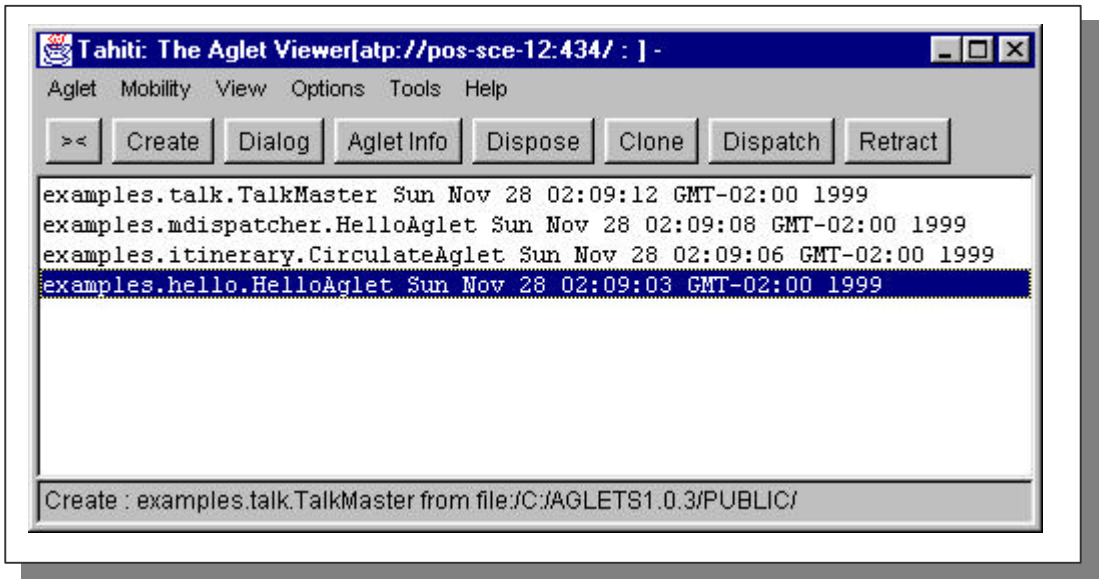

Figura 5-2 - Tahiti: Ambiente de Hospedagem de *Aglets* com 4 agentes em execução

A migração de um *Aglet* inicia-se com a interrupção da execução do *Aglet* na máquina origem, seu envio para uma máquina remota e o reinicio da execução após sua chegada ao destino. Mecanismos de segurança permitem a configuração dos direitos para que *Aglets* não autorizados (*untrusted*) não tenham acesso a determinados recursos do sistema, tornando o sistema implementado com *Aglets* seguro.

A biblioteca de classes de *Aglets* foi concebida pelos pesquisadores da IBM tendo os seguintes objetivos:

- Fornecer um modelo completo e simples de programação para a utilização de agentes móveis, sem no entanto implicar em modificações na máquina virtual Java ou em código nativo;
- Disponibilizar mecanismos de comunicações poderosos e dinâmicos que permitissem agentes comunicarem-se com outros agentes, fossem eles conhecidos ou não;
- Projetar uma arquitetura de agentes móveis que permitisse extensibilidade e reusabilidade;
- Obter uma arquitetura altamente coerente com o modelo tecnológico Web/Java.

Baseado ainda nesta biblioteca, foi desenvolvido o *Fiji kit* que permite criar *applets* na qual executam contextos *Aglets* e podem criar, despachar e receber agentes de páginas Web.

5.3.1 Modelo Aglet

Esse modelo foi desenvolvido para beneficiar as características dos agentes de Java enquanto supera algumas das deficiências da linguagem. No modelo de objeto *Aglet*, um agente móvel é um objeto móvel que tem sua própria *thread* de controle, é orientado a evento e comunica por passagem de mensagem.

Esse modelo define um conjunto de abstrações e o comportamento necessário para alavancar a tecnologia de agentes móveis em redes de longa distância abertas, como a Internet. A abstração chave é composta por: *Aglet*, *proxy*, contexto e identificador.

- ***Aglet*** - Um *Aglet* é um objeto Java móvel que visita *hosts* habilitados em uma rede de computadores.

- ***Proxy*** - Um *proxy* é uma representação de um *Aglet*. Ele serve como uma proteção para o Aglet contra acessos diretos a seus métodos públicos (figura 5-3). O *proxy* também fornece transparência de localização para o *Aglet*; isto é, ele pode ocultar a localização real dos *Aglets*. Isso significa que um *Aglet* e seus *proxies* podem estar separados, de forma que um *proxy* local oculta à distância o *Aglet*.

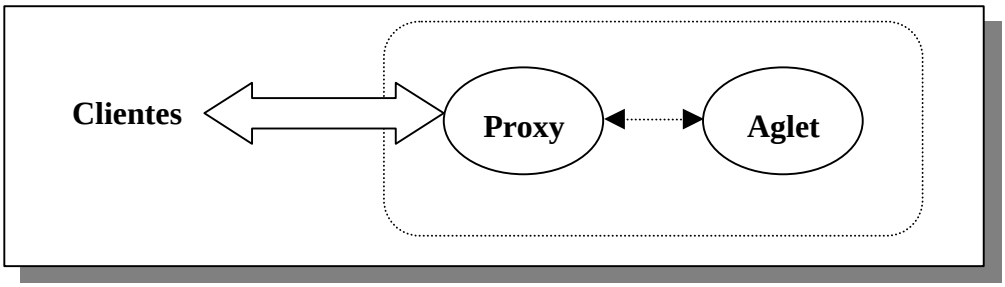

Figura 5-3 - Relacionamento entre *Aglet* e *Proxy* [LANGE & OSHIMA, 1998]

- ***Contexto*** - O contexto é o lugar de trabalho do *Aglet*. Ele é um objeto estático que fornece um meio de manter e gerenciar *Aglets* rodando em um ambiente de execução uniforme onde o sistema *host* é protegido contra *Aglets* maliciosos (figura 5-4). Um nó em uma rede de computadores pode rodar múltiplos processos servidores e cada servidor pode hospedar múltiplos contextos. Contextos são nomeados e assim podem ser localizados pela combinação do endereço do servidor e seu nome.

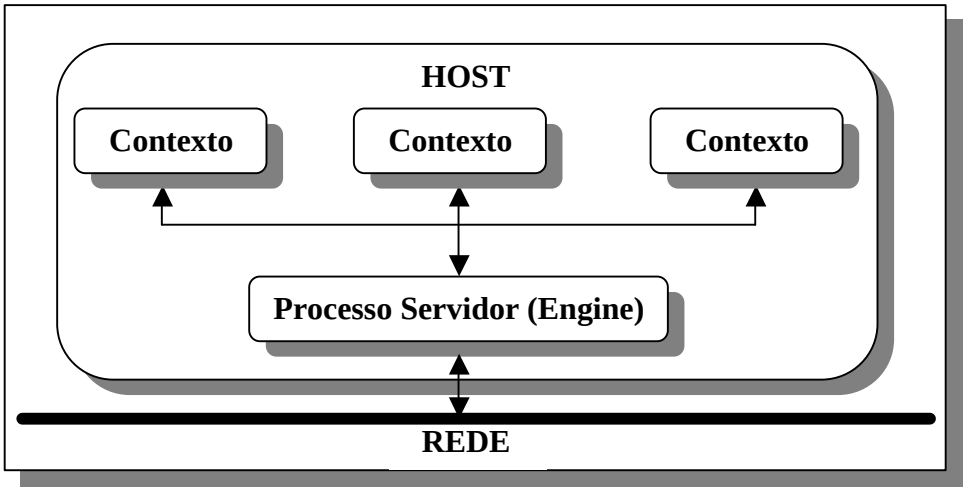

Figura 5-4 - Relacionamento entre Host, Processo Servidor (Engine) e Contextos

- ***Identificador*** - Um identificador é associado para cada *Aglet*. Esse identificador é único e imutável ao longo do ciclo de vida do *Aglet*.

Os comportamentos de um modelo de objeto *Aglet* são baseados em uma cuidadosa análise da vida e morte de agentes móveis. Há somente duas formas de dar vida a um *Aglet*: uma é instanciá-lo no momento de sua criação e outra é copiá-lo de um *Aglet* existente (*clone*). Para controlar a população de *Aglets* pode-se, é claro, destruí-los (*disposal*). *Aglets* são móveis de duas formas: ativa e passiva. Em uma abordagem ativa, eles movem-se por conta própria do *host* corrente para o *host* remoto (*dispatching*). O *host* remoto invocando o *Aglet* do *host* corrente (*retracting*) caracteriza o tipo passivo de mobilidade do *Aglet*. Quando *Aglets* estão em execução, eles consomem recursos. Para reduzir esse consumo, *Aglets* podem ser colocados em modo dormente temporariamente, liberando seus recursos (*deactivation*) e, posteriormente, podem ser colocados novamente no modo de execução (*activation*). Finalmente, múltiplos *Aglets* podem trocar informações para completar uma determinada tarefa (*messaging*).

Este é o conjunto mínimo de operações necessárias para criar e gerenciar um ambiente de agentes móveis distribuído.

A seguir é apresentado as operações fundamentais de um *Aglet*. Estas, são representadas graficamente na figura 5-5:

- ***Creation*** - um novo *Aglet* é associado a um identificador, inserido dentro do contexto e inicializado.

- ***Cloning*** - o *cloning* de um Aglet produz uma cópia quase idêntica do *Aglet* original no mesmo contexto. Somente as diferenças de identificador e o fato que a execução inicializa um novo *Aglet*. *Threads* de execução não são clonadas.

- ***Dispatching*** - despachar um *Aglet* de um contexto para outro irá removê-lo do contexto corrente e inseri-lo no contexto destino, onde ele reiniciará a sua execução (*threads* de execução não migram). Diz-se que o *Aglet* foi enviado para um novo contexto.

- ***Retraction*** - a recuperação de um *Aglet* irá chamá-lo (removê-lo) do contexto corrente e inseri-lo no contexto do qual a chamada foi realizada.

- ***Activation*** e ***deactivation*** - a desativação de um *Aglet* é a habilidade de, temporariamente, parar sua execução e armazenar o seu estado em um dispositivo de armazenamento secundário. A ativação de um *Aglet* irá restaurá-lo no mesmo contexto.

- ***Disposal*** - a liberação de um *Aglet* irá terminar sua execução e removê-lo do contexto corrente.

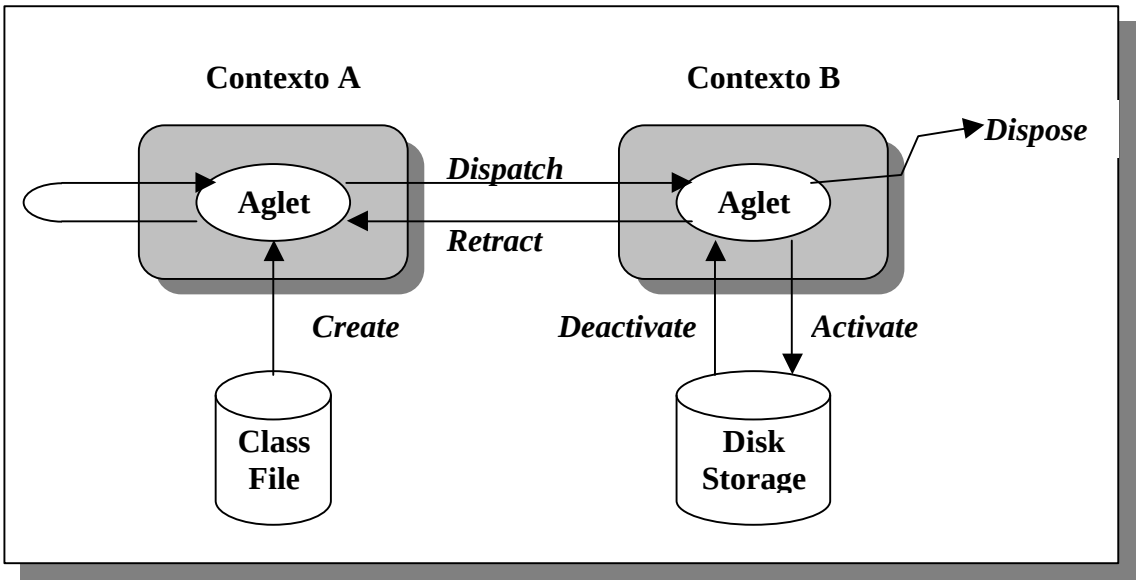

Figura 5-5 - Modelo do Ciclo de vida de um *Aglet*

5.4 Padronização para Agentes Móveis: MASIF

Evidentemente, os ambientes de desenvolvimento de agentes mencionados apresentam grandes diferenças de arquitetura e implementação. Dessa forma, impedem a interoperabilidade e o rápido desenvolvimento da tecnologia de agentes móveis. Para promover essa interoperabilidade, alguns aspectos da tecnologia de agentes móveis precisava ser padronizada. Cinco companhias*: Crystaliz, General Magic Inc, GMD Fokus, IBM Corporation* e o *The Open Group*, apresentaram juntas uma proposta para uma *Facilidade de Interoperabilidade para Sistemas de Agentes Móveis* (MASIF) e esta chamou a atenção do *Object Management Group* (OMG - http://www.omg.org), sendo aceita como padrão OMG em fevereiro de 1998.

Interoperabilidade entre linguagem para objetos móveis é muito difícil e MASIF é direcionado para interoperabilidade entre sistemas de agentes escritos na mesma linguagem, mas

potencialmente por vendedores diferentes, conforme pode ser observado na figura 5-6. Além disso, MASIF não tenta padronizar operações locais de agentes como interpretação de agentes, serialização ou execução. Pode-se dizer que MASIF define a interface para o agente a nível de sistema e não a nível de agente.

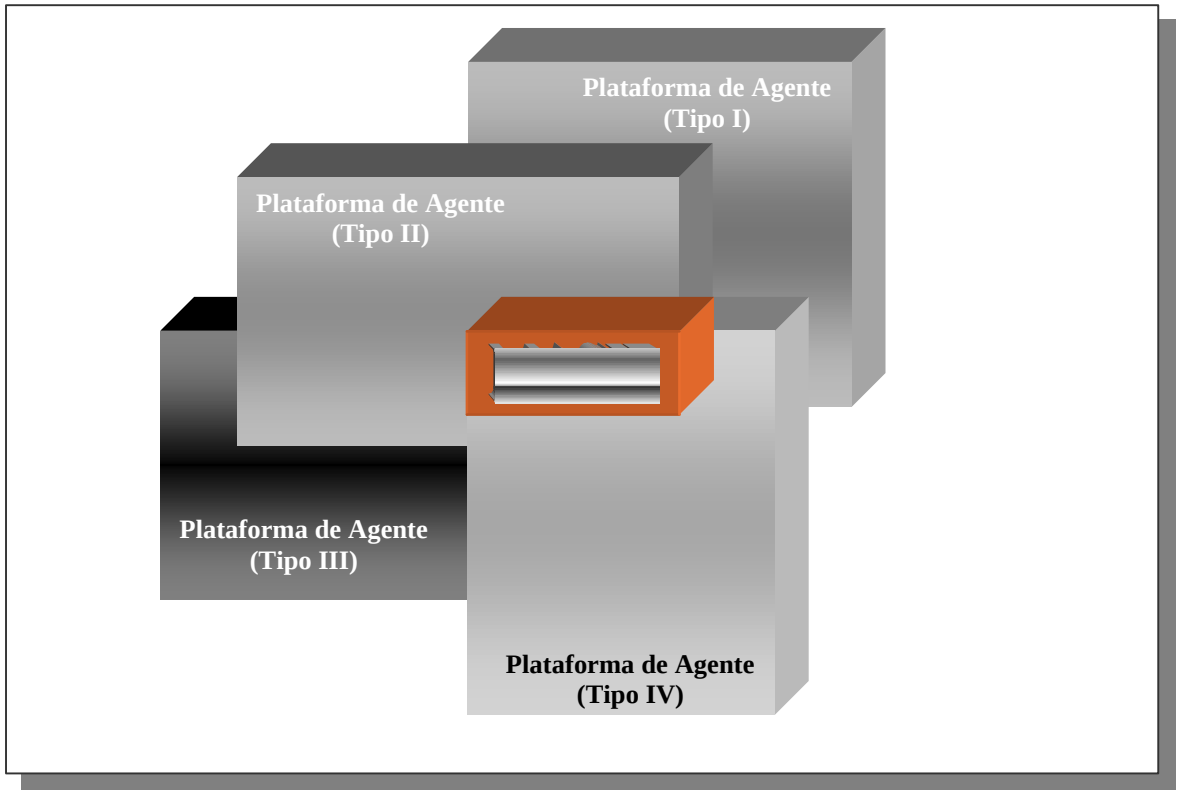

Figura 5-6 - Representação do objetivo do padrão MASIF

Desta forma, MASIF padroniza as seguintes áreas:

- *Gerenciamento de Agentes:* É desejável que o administrador do sistema de um sistema de agente consiga operar um outro através de operações padrões. Isso possibilita encontrar um agentes, criar um agente, suspender sua execução, concluir sua execução ou terminá-la de uma forma padrão.

- *Transferência de Agentes:* É desejável que aplicações agentes possam disparar vários agentes para moverem-se livremente através de sistemas de agentes de diferentes tipos, resultando em uma infra-estrutura comum.

- *Identificação de Agentes e de Sistemas de Agentes:* Em adição à padronização de interoperabilidade entre sistemas de agentes, a sintaxe e semântica de vários parâmetros é padronizada. Especialmente a identificação de agentes e de sistemas de agentes, permitindo a sistemas de agentes e agentes identificarem-se uns aos outros e permitir a aplicações identificarem agentes e sistemas de agentes.

- *Sintaxe de Localização e Tipo de Sistema de Agentes:* A sintaxe de localização é padronizada para permitir agentes acessarem informações sobre o tipo do sistemas agentes de destino . O agente poderá ser enviado somente se o sistema de agente de destino puder suportar o agente. Sintaxe de localização também é padronizada para que sistemas agentes possam localizar uns aos outros.

5.5 Considerações Finais

Como discutido neste capítulo, encontramos em Java um meio de implementação para agentes móveis, uma vez que muitas das características de Java são valiosas para sistemas de agentes móveis. As mais importantes são independência de plataforma, execução segura, carregamento dinâmico de classes, programação *multithreads*, serialização de objetos e introspecção. Algumas vantagens e desvantagens foram apresentadas. A mais significante dessas desvantagens é o suporte inadequado a recursos, falta de proteção de referências, falta de um objeto proprietário de referências e falta de suporte para preservação e recuperação do estado de execução. Algumas dessas desvantagens podem ser trabalhadas, outras são mais sérias e tem implicações em todo o conceito e desenvolvimento de sistemas de agentes móveis baseados em Java.

Dentre as diversas plataformas de desenvolvimento de agentes móveis apresentadas, foi escolhida a plataforma *Aglets*, sendo seu modelo apresentado e discutido. Esta escolha deve-se principalmente ao fato das configurações de segurança oferecidas por este ambiente. No modelo *Aglets*, um agente móvel tem sua própria *thread* de controle, é dirigido a evento e estabelece comunicação por passagem de mensagem.

Uma vez que os diversos sistemas de agentes móveis baseados em Java possuem muito em comum, mas não apresentam interoperabilidade, foi apresentado a padronização aceita pela

OMG: MASIF. A utilização desta padronização determinará um impacto crescente no uso da tecnologia de agentes móveis.

A seguir é apresentado os sites onde se pode obter maiores informações sobre os principais projetos de agentes móveis discutidos neste capítulo:

- Aglets: http://www.trl.ibm.co.jp/aglets;
- Odyssey: http://www.generalmagic.com;
- Concordia: http//www.meitca.com/HSL/Projects/Concordia;
- Voyager: http://www.objectspace.com/voyager;
- Agent Tcl: http://www.cs.darthmouth.edu/~agent;
- Ara: http://www.uni-kl.de/AG-Nehmer/Projekte/Ara;
- TACOMA: http://www.cs.uit.no/DOS/Tacoma;

6 Implementação

6.1 Considerações iniciais

Este capítulo apresenta, inicialmente, os métodos necessários para o desenvolvimento de um agente móvel simples utilizando-se do conceito de *Aglets.* Esta implementação visa demonstrar a estrutura de um agente *Aglet* para uma melhor compreensão do cenário implementado.

A seguir, é apresentado uma breve discussão sobre a implementação dos agentes que compõem o cenário de Identificação de Usuários Anômalos.

6.2 Entendendo um Aglet Simples

Para uma melhor compreensão da implementação dos agentes contidos neste trabalho, é necessário que se entenda a estrutura básica de um *Aglet*.

Um *Aglet* não é instanciando como um objeto Java qualquer. Todo *Aglet* é criado dentro de um *contexto de agentes* em um servidor de *Aglets*, no nosso caso o próprio TAHITI. O *Aglet* é sujeito às regras e restrições impostas pelo contexto no qual foi criado. Para que o *Aglet* consiga comunicar com contextos, deve-se criar uma interface entre o *Aglet* e o contexto. Para isso, existe um método no *Aglet* chamado de *getAgletContext*, que retorna uma interface *AgletContext.* Essa interface é utilizada pelo *Aglet* para capturar as informações sobre o ambiente e também, para enviar mensagens para o ambiente e outros *Aglets* ativos no ambiente.

Quando um *Aglet* é criado pelo contexto, são invocados três métodos:

- O método construtor do *Aglet*;
- O método *onCreation*, que é executado assim que o método construtor finaliza o seu trabalho;

- E por último, é chamado o método *run***,** que inicia a execução do *Aglet.*

Além desses três métodos básicos, o *Aglet* possui outros que são importantes no seu funcionamento. Alguns deles são:

- Método ***dispose***¸ utilizado para "destruir" o *Aglet*;
- Método ***onDisposing,*** chamado quando é executado o método ***dispose***, muito utilizado para liberar algum recurso alocado pelo *Aglet* antes de sua "destruição";
- E o método ***handleMessage,*** que é acionado quando o *A glet* recebe alguma mensagem do contexto ou outro *Aglet.*

A seguir, é apresentado o código fonte de um *Aglet* bem simples que demonstra a utilização dos métodos descritos anteriormente. Todos os métodos básicos de um *Aglet* são encontrados no pacote **com.ibm.aglet**. A função deste *Aglet* é escrever no console do sistema o nome do método que ele está executando.

```
/** Criando um aglet simples **/
package examples.estudo;
import com.ibm.aglet.*; //Bibliotecas básicas do aglet
public class AgenteSimples extends Aglet { // herdando características de Aglets
/* Método construtor */
public AgenteSimples() {
System.out.println("AgenteSimples: Estou no método construtor!");
}
/* Método executado automaticamente após o construtor */
public void onCreation(Object init) {
System.out.println("onCreation: Acabei de ser criado!");
}
/* Método executado automaticamente após o dispose */
public void onDisposing() {
            System.out.println("onDisposing: Estou liberando os recursos
alocados!");
}
/* Método inicial de execução de um aglet*/
public void run() {
System.out.println("run: Iniciando a execução");
try {
   /* Envia uma mensagem */
   getProxy().sendMessage( new Message("alomundao"));
}
catch (Exception ex) {
       System.out.println("Erro: "+ex);
}
}
/* Método ativado quando o aglet recebe uma mensagem */
public boolean handleMessage(Message msg) {
if (msg.sameKind("alomundao")) { //Verificando a mensagem recebida
 System.out.println("Alomundao");
} else return false;
return true;
}
```

No código anterior, o método ***sendMessage*** é utilizado para enviar uma mensagem para um *Aglet.* Esse, pode ser outro *Aglet* qualquer ou ele mesmo. Nesse caso, o *Aglet* está enviando uma mensagem para ele mesmo, pedindo para que seja apresentado a mensagem "Alomundao" no console do sistema. A Figura 6-1 apresenta como fica o console após a execução desse *Aglet.*

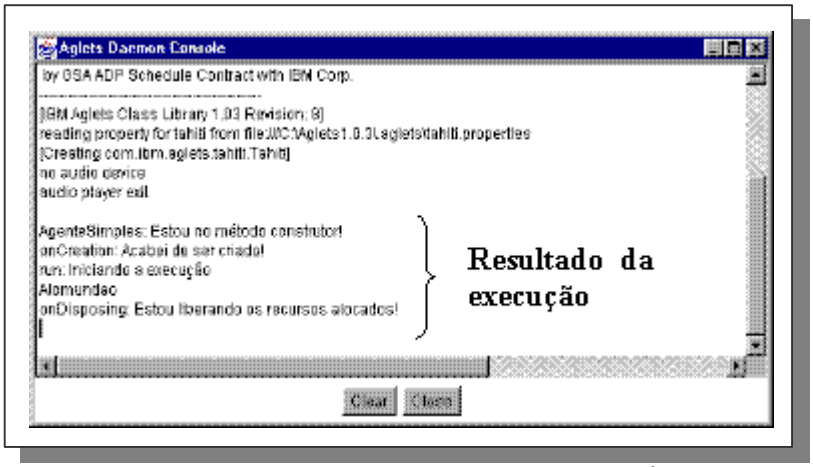

Figura 6-1– Console com resultado da execução do *Aglet*

6.3 Implementação do Cenário de Identificação de usuários anômalos

O cenário escolhido para apresentação neste trabalho, foi o de Identificação de usuários anômalos. Este cenário e composto, inicialmente, por 5 agentes dispostos da seguinte forma:

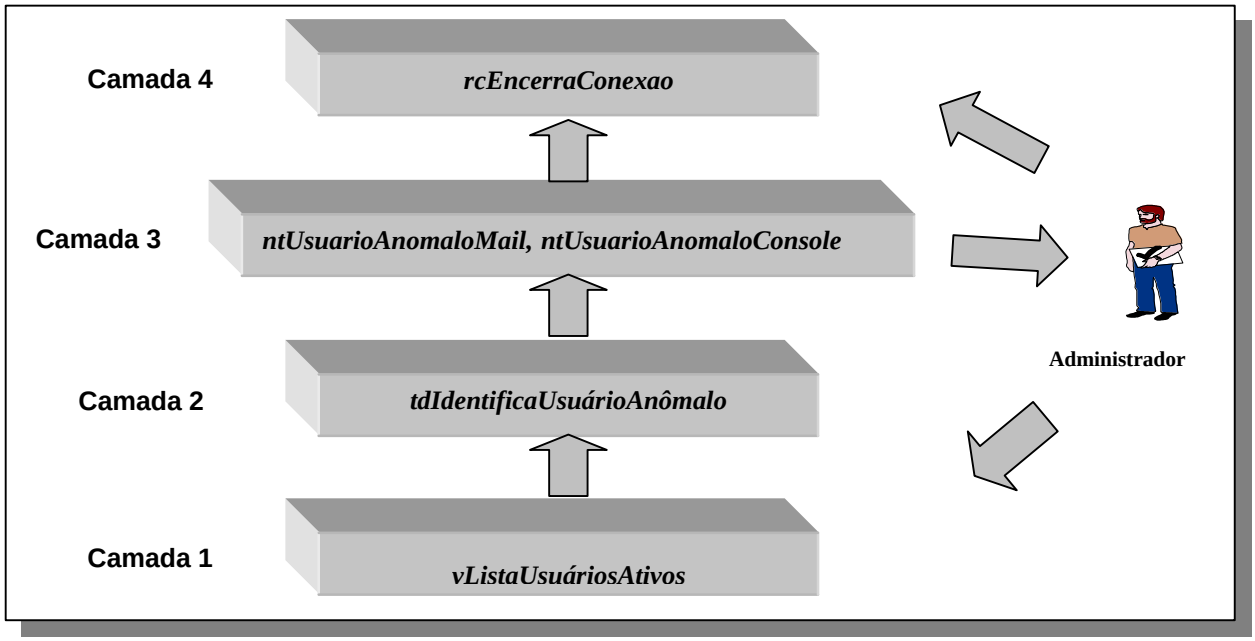

Figura 6-2 - Representação dos Agentes do Cenário de Identificação de usuários anômalos

6.4 Agente da Camada de Vigilância

Agente de Vigilância: *vListaUsuáriosAtivos*
Parâmetro: IP da máquina a ser monitorada;
Função: Captura de nomes de usuários "conectados" ao servidor;
Agente ativador: Agente de Controle ou Administrador;
Agente a ser ativado: *tdIdentificaUsuarioAnomalo*.

O agente de vigilância tem a função de "viajar" pelas máquinas que irá monitorar e capturar um lista com informações dos usuários. Exemplificando, se o sistema operacional da máquina de destino for padrão UNIX, o agente executará os comando *w* e *who*, obtendo informações dos usuários que estão "conectados" ao sistema. Ao retornar, ele ativa o agente de tomada de decisão enviando estas informações em uma mensagem.

O agente de vigilância pode ser ativado tanto pelo agente de controle do sistema , figura 6-3(a), quanto pelo Administrador, figura 6-3(b).

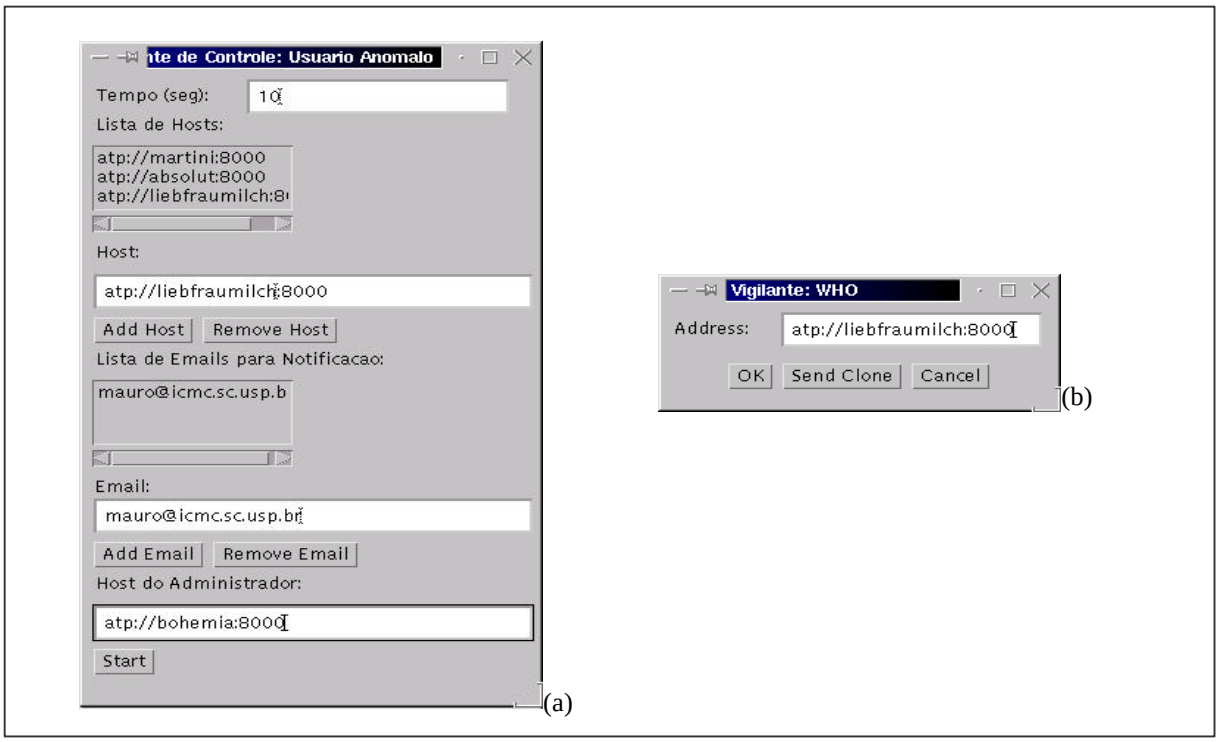

Figura 6-3 – Telas Para Ativação: (a)Agente de Controle e (b)Administrador.

A seguir, são apresentados os trechos de código mais importantes deste agente.

```
/* Executa o comando e atribui um processo */
pExecutaComando=Runtime.getRuntime().exec(“who”);
/* Atribui o stream de resultado ao stream gerado pelo processo */
disResultadoComando = new
DataInputStream(pExecutaComando.getInputStream());
sResultadoProcessado = "";
/* Transfere o que esta no Stream de resultado para uma string */
/* esta string possui o resultado da execução do comando WHO */
while ((sLinhaResultado = disResultadoComando.readLine())!= null) {
              sResultadoProcessado += sLinhaResultado + "\n\r";
}
```

Observamos no trecho anterior a instrução `itinerary.go(destination, "execWho").` O método go da classe `itinerary` é responsável por enviar (dispatch) o *Aglet* para o servidor a ser monitorado. O endereço de destino está em `destination` e foi recebido da tela de ativação do agente ou do agente de controle. Esse método envia também a mensagem “`execWho`”. Essa, irá acionar o método de execução do comando `Who`. Esse, irá capturar informações das conexões ativas.

```
SimpleItinerary itinerary = null;

/**** trecho responsável por enviar o agente ao
servidor destino
        try {
            itinerary.go(destination, "execWho");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
/******
         try {
          itinerary.go(home, "atHome");
        } catch (Exception ex) {
            ex.printStackTrace();
}
```

O trecho `itinerary.go(home, "atHome");` contém as informações para, após a execução do agente no servidor remoto, envia-lo de volta o servidor de origem. A mensagem `atHome` será responsável pela ativação do agente tomada de Decisão e a desativação (dispose) do agente de vigilância.

```
public boolean action(Event ev, Object obj) {
        if (ev.target == send) {
            aglet.message = msg.getText();
           try {
              /** criação do clone do Aglet para ser enviado ao servidor*/
              AgletProxy aglet2 = (AgletProxy)aglet.clone();
aglet2.sendOnewayMessage(new message("inicia",address.getText()));
           } catch (Exception e) {
              e.printStackTrace();
           }
      } else if (ev.target == go) {
            aglet.message = msg.getText();
             /*envia uma mensagem para disparar o agente p/ servidor remoto*/
            aglet.handleMessage(new Message("inicia", address.getText()));
        } else if (ev.target == close) {
            hide();
        } else {
            return false;
        }
        return true;
    }
}
```

O trecho de código anterior apresenta o trecho de envio (*dispach*) do agente de vigilância.

6.5 Agente da Camada de Tomada de Decisão

Agente de Tomada de Decisão: *tdIdentificaUsuárioAnômalo*

Parâmetro: Informações coletadas pelo agente de vigilância;

Função: Verificar a existência de usuários anômalos;

Agente ativador: Agente de Vigilância

Agente a ser ativado: Agente de Notificação para envio de E-mail *ntUsuárioAnômaloMail* e agente de notificação via console *ntUsuárioAnomaloConsole*

O agente dessa camada (*tdIdentificaUsuarioAnomalo)* é ativado pelo agente da camada de Verificação (*vListaUsuariosAtivos)* recebendo como parâmetros uma lista contendo informações dos usuários conectados ao sistema. Esse agente irá, com base em um arquivo de *profile*, verificar possíveis usuários anômalos. Ao suspeitar de uma possível anomalia, este agente irá invocar os agentes de notificação da camada superior.

6.6 Agentes da Camada de Notificação

Neste cenário, essa camada é composta inicialmente por 2 agentes. Estes agentes, com base nas informações recebidas do agente de tomada de decisão, irão notificar o administrador da rede da suspeita de um usuário anômalo.

6.6.1 Agente de notificação por e-mail

Agente de Notificação: *ntUsuarioAnomaloMail* (Agente Estático)
Parâmetro: e-mail do adminstrador;
Função: enviar e-mail ao administrador identificando usuário anômalo;
Agente ativador: *tdIdentificaUsuarioAnomalo*
Agente a ser ativado: *rcEncerraConexão*

Este agente é responsável pela notificação via e-mail e acionamento do agente de reação da camada superior. A seguir, é apresentado o trecho principal de código desse agente.

```
try {
   /* Cria um socket para a porta de SendMail do servidor de email */
   Socket sendMail = new Socket("icmc.sc.usp.br", 25);
   /* Cria um stream para escrever a mensagem no socket */
   PrintStream mensagem = new PrintStream(sendMail.getOutputStream());
   /* Cria um stream para ler as mensagens do socket */
   DataInputStream in = new DataInputStream(sendMail.getInputStream());
   /* Escrever a mensagem */
   mensagem.println("HELO "+hostEnvio);
   System.out.println(in.readLine()); //Lendo a resposta do servidor
   mensagem.println("MAIL FROM: "+emailAdmin);
   System.out.println(in.readLine());
   mensagem.println("RCPT TO: "+emailAdmin);
   System.out.println(in.readLine());
   mensagem.println("DATA");
   System.out.println(in.readLine());
   mensagem.println("Subject: Notificação de ususário anômalo");
   mensagem.println("Atencão às "+horaDetect+" foi detectado uma anomalia para o
"+"usuário "+usuario);
   mensagem.println("Este, se conectou ao sistema às "+horaConnect+" através do
"+"endereço "+ipUser);
   mensagem.println("Este usuário foi bloqueado às "+horaDetect);
   mensagem.println("Favor verificar arquivos de LOGS!");
   mensagem.println("Agente de Notificação via Email");
   mensagem.println("\n\r");
   mensagem.println(".");
   mensagem.println("\n\r");
   System.out.println(in.readLine());
   mensagem.println("QUIT");
   System.out.println(in.readLine());
   mensagem.flush();
   sendMail.close(); ...}
```

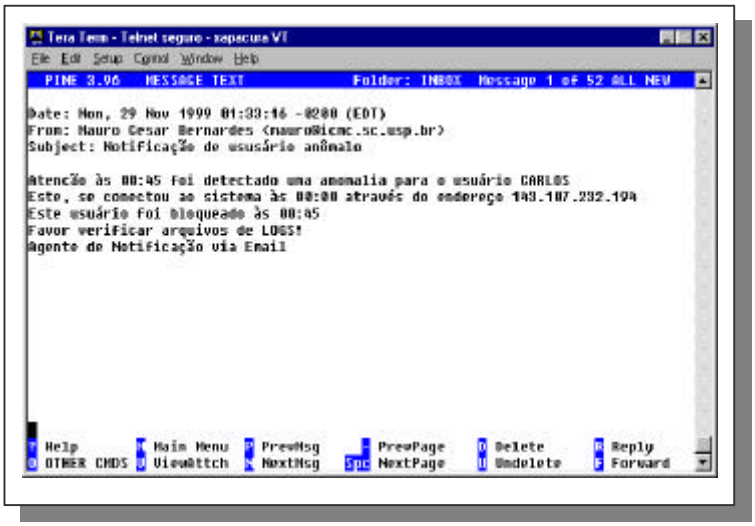

Figura 6-4 - Notificação de Uma Anomalia Recebida via e-mail

A figura 6-4 apresenta a notificação de uma suspeita de intrusão com base em anomalias de usuários recebida pelo administrador do sistema via e-mail. A figura 6-5 apresenta as mensagens de console para o acionamento do agente.

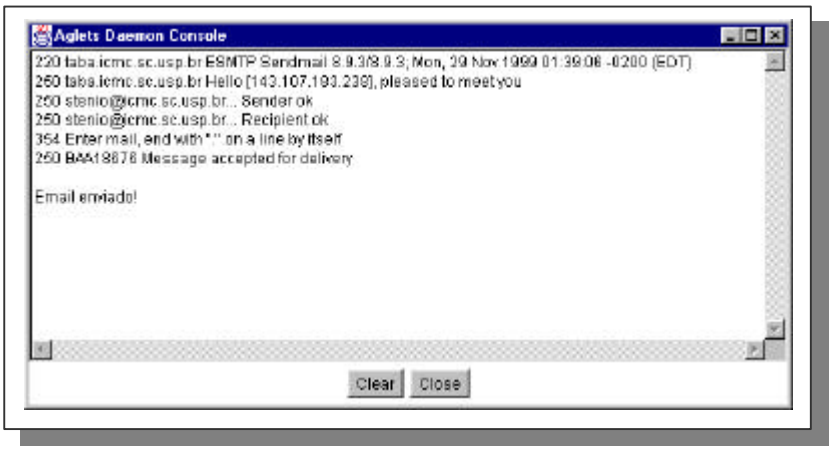

Figura 6-5 - Console de Execução para o Agente de Notificação via e-mail

6.6.2 Agente de notificação via Console

Agente de Notificação: *ntUsuarioAnomaloConsole*
Parâmetro: IP da máquina do admistrador;
Função: notificar o administrador de usuários anômalos via console
Agente ativador: *tdIdentificaUsuarioAnomalo*;
Agente a ser ativado: *Nenhum*

Este agente é responsável pela notificação do administrador do sistema enviando uma mensagem para o console da máquina ao qual ele conectado.

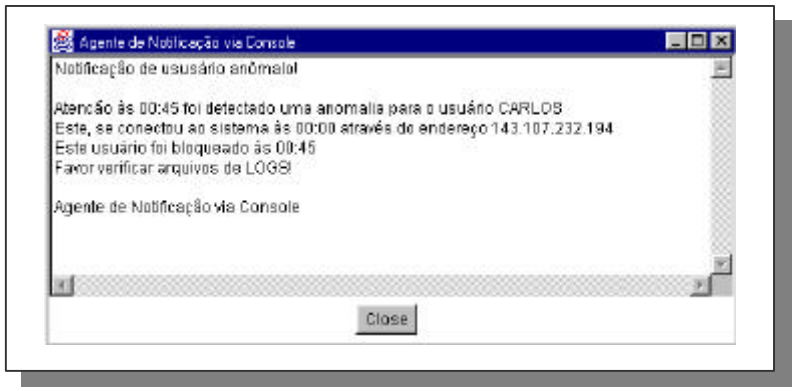

Figura 6-6 - Notificação via console de uma Suspeita de Anomalia

6.7 Agente da Camada de Reação

Neste cenário, este agente é responsável por encerrar e bloquear a conexão para um usuário identificado como anômalo.

Seu código é muito semelhante ao código do agente de vigilância deste cenário e compreende a execução de um comando *kill* seguido de um bloqueio da senha do usuário.

Agente de Reação: *rcEncerraConexão*
Parâmetro: Informação do usuário anômalo
Função: Derrubar conexão e bloquear acesso deste usuário
Agente ativador: *ntUsuarioAnomaloMail*
Agente a ser ativado: *nenhum*

6.8 Considerações Finais

Este capítulo apresentou a descrição de implementação do cenário de Identificação de usuários anômalos, modelado na seção 4.5.1. Apesar de constituir o cenário mais simples dentre os modelados, este consegue expressar claramente o processo de comunicação entre as camadas do modelo proposto no capítulo 4.

Aproveitando este cenário inicial, novos agentes podem ser desenvolvidos e inseridos em cada camada, o que vem demonstrar a escalabilidade do sistema proposto. Como exemplo, pode-

se inserir novos agentes na camada de verificação, aumentando seu grau de especialização, sem a necessidade de alteração nos agentes das camadas superior. O mesmo pode acontecer nas demais camadas.

7 Conclusões e Trabalhos Futuros

Com a crescente utilização das redes de computadores e sua conseqüente abertura para um mundo externo, os sistemas computacionais tornaram-se cada vez mais difíceis de serem protegidos. Assim, uma das grandes preocupações dos administradores desses sistemas era criar barreiras de proteção contra invasores externos.

Entretanto, pesquisas recentes demonstram que a maioria dos ataques é proveniente internamente da própria empresa ou organização, sendo proferidos por funcionários ou similares e alcançando números da ordem de 70% (http://www.gocsi.com/). Os demais, são procedentes de fora da organização. Neste caso, os ataques provêm, em geral, da Internet.

Dessa forma, não basta simplesmente criar barreiras de proteção contra um mundo externo, como é o caso dos *firewalls*. Sistemas de Detecção de Intrusão (SDI) devem estar presentes e em constante monitoria, buscando informações que possam identificar além de atacantes externos, usuários internos fazendo mal uso de seus privilégios (de forma intencional ou acidental).

A grande dificuldade encontrada na detecção de uma intrusão interna é que ela é móvel, proveniente de diversos pontos da rede interna. Neste caso, os Sistemas de Detecção de Intrusão tradicionais (*host based, network based* ou híbridos) devem ser alocados em diversos pontos estratégicos, em busca da identificação de uma intrusão ou tentativa de ataque no momento em que ela aconteça.

Em um trabalho anterior, Elderclei Regis Reami [REAMI, 1998] apresenta a especificação de requisitos e prototipagem de um ambiente de gerenciamento de segurança apoiado por agentes móveis. A especificação apresentada define as diretrizes para o desenvolvimento de um ambiente de segurança com características de tolerância a falhas.

Este trabalho apresenta-se como uma continuidade do trabalho anterior. A definição de uma arquitetura de comunicação entre agentes para o desenvolvimento de SDI não monolítico e com características de mobilidade é apresentado, utilizando-se de um modelo em camadas. Esse modelo representa um padrão significativo para o sistema, provendo facilidades de

escalabilidade, manutenção e desenvolvimento de cenários de segurança. Cada camada determina os limites de atuação de um agente, criando uma estrutura de comunicação clara que permite a inserção e remoção de novos agentes dentro de um cenário ou ainda, o compartilhamento de agentes entre cenários distintos.

Outras contribuições deste trabalho são:

- Identificação das vantagens de um SDI não-monolítico e móvel sob um SDI monolítico;
- A apresentação de uma proposta de modelagem para compreensão e posterior desenvolvimento de novos cenários com características de segurança;
- A implementação de um cenário para a composição de um Sistema de Detecção de Intrusão com características híbridas da forma de detectar intrusão e em termos do tratamento dos dados para a detecção. O cenário implementado consiste de um conjunto de agentes móveis que desempenharão funções de monitora, tomada de decisão, notificação e reação (contra-ataque) para comportamentos anômalos de usuários no sistema.

Com a crescente facilidade de se obter informações sobre quebras de segurança e o lançamento de modernas técnicas e sua divulgação pela Internet, torna-se necessário o desenvolvimento constante de novas técnicas de proteção. Isso reforça uma das grandes virtudes do Sistema Proposto: sua escalabilidade, facilitada pela capacidade de inserção de novos agentes móveis ou cenários à medida que novos conhecimentos de proteção e prevenção são adquiridos.

Assim, este trabalho apresenta uma proposta para um Sistema de Detecção de Intrusão não-monolítico, móvel, escalável, tolerante a falhas, independente de plataforma e de fácil manutenibilidade.

7.1 Trabalhos futuros

Como trabalhos futuros para a presente pesquisa, é proposto:

- Desenvolver um estudo aprofundado testando a performance e conseqüente degradação do ambiente com a inserção de agentes móveis em novos cenários. Desta forma, tentar definir a carga de trabalho máxima para cada agentes em cada cenário.

- Analisar os ambientes servidores de agentes móveis existentes. Durante a análise, procurar identificar os que estão em conformidade com a proposta de padronização MASIF e proceder com uma avaliação de performance, facilidade de utilização, degradação do ambiente, interoperabilidade, requisitos de segurança fornecidos, tolerância a falhas, etc. Esse estudo além de definir uma plataforma ideal para utilização em segurança computacional poderá elaborar uma análise de requisitos e posterior especificação para o desenvolvimento de uma plataforma específica para um ambiente de segurança.

- Associar o sistema desenvolvido a um sistema de gerencia, possivelmente o Netracker [MOURO, 1997], de forma a utilizar as informações coletadas pelos agentes para prover estatísticas de utilização da rede e facilidades de administração.

- Definir, modelar e implementar novos cenários de segurança computacional.

- Introduzir o conceito de algoritmos genéticos e redes neurais para a criação de módulos inteligentes de tomada de decisão, provendo cenários avançados de detecção de intrusão.

- Estudar ambientes como o FIJI IBM para prover gerenciamento e ativação de *Aglets* através de contextos formados por *applets* Java.

- Modelar uma base de dados a partir das informações coletadas pelos agentes para a composição de um Sistema de Informação. As informações contidas neste sistema ajudarão, entre tantas coisas, no entendimento do cenário empresarial

- Desenvolver uma linguagem de comunicação segura entre agentes, utilizando de métodos criptográficos e avaliar formas de comunicação: quando disparar agentes e quando disparar mensagens.

Referências Bibliográficas

[BARRUS & ROWE 1998] BARRUS, J.; ROWE, N.C. *A Distributed Automous-Agent Network-Intrusion Detection and Response System*. In: Proceedings of the 1998 Command and Control Research and Technology. Monterrey CA, Junho-Julho 1998.

[BEGUELIN, 1994] BEGUELIN, A., *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press, 1994.

[BELLOVIN & CHESWICK, 1994] BELLOVIN, S.; CHESWICK, W. *Firewalls and Internet Security*: Repelling the Wily Hacker. Addison-Wesley. 1994.

[BELLOVINS, 1989] BELLOVINS, S. M. *Security Problems in the TCP/IP Protocol Suite.* Computer Communications Review, 9(2):32-48, abr. 1989. Disponível on-line em: http://www.cert.lu/security/documents.html. Visitado em 15/12/1998.

[BONIFÁCIO, 1998] BONIFÁCIO Jr., J. M. *Sistemas de Segurança Distribuído: Integração de Firewalls com Sistemas de Detecção de Intrusão.* São Carlos, 1998. Tese (Mestrado) - Instituto de Ciências Matemáticas e de Computação de São Carlos, Universidade de São Paulo.

[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 Inteligence and Multimedia Applicatio*ns*, 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 Simposyum III*, p.63-76. Setembro 1992. Disponível on-line em: http://www.cert.lu/security/documents.html. Visitado em 17/11/1998.

[CHESS et al, 1995] CHESS, David; HARRISON, Colin; KERSHENBAUM, Aaron. *Mobile Agentes: Are They a Good Idea?* IBM Research Report. Disponível on line em: http://www.research.ibm.com/iagentes/paps/mobile-idea.ps. Visitado em 15/01/1999.

[CICILIANI, 1994] CICILIANI, 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.; GARIJO, 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. Disponível on-line em: 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). Disponível on-line em: http://www.cs.purdue.edu/homes/spaf/tech-reps/9522.ps. Visitado em 15/01/1999.

[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). Disponível on-line em: http://www.cs.purdue .edu/homes/spaf/tech-reps/9508.ps. Visitado em 15/01/1999.

[DARPA 1997] DEFENSE ADVANCED RESEARCH PROJECTS AGENCY. *Adaptive Systems Security Policies*. 1997. Disponível on-line em: http://www.darpa.mil/ito/Summaries97/F256_0.html. Visitado em 08/10/1998.

[ENDLER, 1998] ENDLER, Markus. *Novos Paradigmas de Interação usando Agentes Móveis.* Departamento de Ciência da Computação. IME. USP. SBRC98. Disponível on-line em http://www.ime.usp.br/~endler/paperlinks/sbrcslides.ps. Visitado em 25/03/1999.

[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. Disponível on-line em: http://www.msci.memphis.edu /~franklin/AgentProg.html. Visitado em 05/12/1998.

[FERREIRA , 1977] FERREIRA, Aurélio. B. H. *Minidicionário Aurélio da Língua Portuguesa.* 1ª edição, 16ª impressão. Editora Nova Fronteira, 1977.

[FUGGETA et al., 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.

[GEIST, 1994] GEIST, A., Beguelin, A., Dongarra J., Jiang, W., Manchek, R., Sunderam V., *PVM 3 User's Guide and Reference Manual*, Oak National Laboratory, September, 1994.

[GENESERETH & KETCHPEL, 1994] GENESERETH, M.R.; KETCHPEL, S.P. *Software Agents.* Communications of the ACM, 37(7): 48-53, 1994.

[GANE, 1988] GANE, Chris.*Desenvolvimento rápido de sistemas.* Rio de Janeiro: LTC-Livros Técnicos e Científicos Editora, 1988

[GANE & SARSON, 1994] GANE, Chris; SARSON, Trish. *Análise Estruturada de Sistemas*. Rio de Janeiro: LTC-Livros Técnicos e Científicos Editora, 1994.

[ISO, 1989] International Organization for Standardization / International Eletrotechnical Committee. *"Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture*". *International Standard 7498-2, 1989.*

[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.

[KOSTER, 1994] KOSTER, Martijn. *The Web Robots Page.* June 1994. Disponível on-line em http://info.webcrawler.com/mak/projects/robots/robots.html. Visitado em 15/04/1999.

[LANGE & OSHIMA, 1998] LANGE, D.B; OSHIMA, M. *Programming And Deploying Java Mobile Agents with Aglets.* Addison Wesley Longman, Inc. 1998.

[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. Visitado em 13/12/1999.

[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.

[LINGNAU & DROBNIK, 1996] LINGNAU, Anselm; DROBNIK, Oswald. *An Infraestructure for Mobile Agentes: Requeriments and Architecture.* Frankfurt am Main, Germany. http://www.tm.informatik.uni-frankfurt.de/ma/paper.html. Visitado em 28/01/1999.

[LUNT & JAGANNATAHAN, 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.

[MAGIC, 1998] GENERAL MAGIC INC. *Odyssey*. Disponível on-line em: http://www.genmagic.com/odyssey/agents, visitado em 15/12/1998.

[MINSK & RIECKEN, 1994] MINSK, M.; RIECKEN, D. *A conversation with Marvin Minsk about Agents.* Communications of the ACM, 37(7):23-29, 1994.

[MITSUBISHI, 1997] MITSUBISHI ELETRIC INFORMATION CENTER. *Concordia - Java mobile Agent Technology*. Disponível on-line em http://www.meitca.com/HSL/Projects/condordia, visitado em 11/11/1999.

[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 Ciências 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. Disponível em WWW no endereço: http://www.info.fundp.ac.be/~amo/publication.html. Visitado em 22/11/1998.

[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.

[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.

[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. Disponível on-line em: http://www.labs.bt.com/projects/agents/publish/papers/sat_report.htm

[NWANA,1996] NWANA, H.S. *Software Agents: An Overview*. In: Knowledge Engineering Review, vol. 11, no. 3, p205-244, Outubro/Novembro 1996.

[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.

[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. Disponível on-line em: http://www.csl.sri.com/emerald/emerald-niss97.html

[PRESSMAN, 1995] PRESSMAN, Roger S. *Engenharia de Software*. São Paulo: Makron Books, 1995.

[REAMI, 1998] REAMI, E. R. *Especificação e Prototipagem de um Ambiente de Gerenciamento de Segurança Apoiado por Agentes Móveis.* São Carlos, 1998, 82p., Dissertação (Mestrado) – Instituto de Ciências Matemáticas de Computação de São Carlos, Universidade de São Paulo.

[RODRIGUEZ, 1999] RODRIGUEZ, Eduardo José. *Uma Modelagem para Comércio Eletrônico usando Corba e Agentes Móveis.* Campinas, 1999, 83p. Dissertação (Mestrado) – Instituto de Computação. Universidade Estadual de Campinas-UNICAMP.

[SANDER & TSCHUDIN, 1998] SANDER, T.; TSCHUDIN, C.F. Towards *Mobile Criptography. In: Proceedings of IEEE Security & Privacy '98*. Oakland, California, Maio 1998. Dosponível on-line em: http://www.icsi.berkeley.edu/~sander/publications/satschu.ps

[SSL, 1996] *SSL 3.0 Specification*. Disponível em WWW no endereço: http://home.netscape.com/eng/ssl3/index.html

[SELKER, 1994] SELKER, T. *Coach: A Teaching Agent that Learns*. Communications of the ACM, 37(7):92-99. 1994.

[SET, 1999] SET, *Secure Eletronic Transaction at Visa*. Disponível em Disponível on-line em: http://www.visa.com/cgi-bin/vee/nt/ecomm/set/main.html

[SOARES, 1995] SOARES, L. F. G., LEMOS, G., COLCHER, S. *Redes de Computadores: das LANs, MANs e WANs às redes ATM.* 2. Edição. Rio de Janeiro: Campus, 1995. 704 p.

[SPACE, 1998] OBJECT SPACE Inc. *ObjectSpace Voyager Overview.* Disponível on-line em http://www.objectspace.com/products/vgrOverview.htm.

[SUN, 1997] SUN MICROSYSTEMS. *Java Beans API specification*. Version 1.01, 1997. Disponível on-line em: http://www.java.sun.com/beans

[TANEMBAUM, 1997] TANEMBAUM, Andrew S. *Computer Networks* Third Edition. Prentice-Hall, Inc, 1997, 923 p.

[VITEK et al, 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. Disponível on-line em: http://cuiwww.unige.ch/OSG/people/jvitek/Publications/tpi.ps.gz

[WACK & CARNAHAN, 1994] WACK, J. P.; CARNAHAN, L. Keeping Your Site Comfortably *Secure: An Introduction to Internet Firewalls*. In: National Institute of Standards and Technology Publication. 1994. WWW: http://www.raptor.com/lib/index.html

[WALKER, 1994] WALKER, D. W., *"The design of a standard message passing interface for distributed memory concurrent computers"*, *Parallel Computing*, vol. 20, pp. 657-673, 1994.

[ZAMBONI et al, 1998] ZAMBONI, Diego; BALASUBRAMANIYAN, Jai; GARCIA-FERNANDES, Jose Omar and SPAFFORD, E. H.; Department of Computer Sciences, Purdue University; Coast TR 98-05; 1998. Disponível on-line em: http://www.cerias.purdue.edu/coast/projects/aafid.html, visitado em 13/01/1999.

[ZERKLE & LEVIT, 1996] ZERKLE, Dan; LEVITT, Karl. *NetKuang - A Multi-Host Configuration Vulnerability Checker.* Departament of Computer Science. University of California at Davis. Disponível on-line em: http://seclabs.cs.ucdavis.edu/papers/zl96.ps, visitado em 13/01/1999.