UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

AGENTES INTELIGENTES PARA DETECÇÃO DE INTRUSOS EM REDES DE COMPUTADORES

CHRISTIANE FERREIRA LEMOS LIMA

São Luis
2002

AGENTES INTELIGENTES PARA DETECÇÃO DE INTRUSOS EM REDES DE COMPUTADORES

Dissertação de Mestrado submetida à Coordenação do Curso de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos para obtenção do título de mestre em Ciência da Computação.

Por

CHRISTIANE FERREIRA LEMOS LIMA

Maio, 2002

Lima, Christiane Ferreira Lemos

Agentes inteligentes para detecção de intrusos em redes de computadores/ Christiane Ferreira Lemos Lima. ___ São Luís, UFMA 2002

177f.:il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal, 2002.

1. Detecção de infecções – Computador. 2. Segurança de Rede. I. Título

CDD 001.642

CDU 004.492.3

AGENTES INTELIGENTES PARA DETECÇÃO DE INTRUSOS EM REDES DE COMPUTADORES

MESTRADO

Área de Concentração: CIÊNCIA DA COMPUTAÇÃO

CHRISTIANE FERREIRA LEMOS LIMA

Orientador: Dr. Edson Nascimento

Curso de Pós-Graduação

Em Engenharia de Eletricidade da

Universidade Federal do Maranhão

Aos meus pais, Raimundo e Marinalva.
Aos meus filhos Arthur e Thaís.
A Orlando, meu esposo.

AGRADECIMENTOS

Muitas pessoas contribuiram para a realização desta pesquisa. Para relacioná-las todas, certamente muitas páginas a mais seriam necessárias. Portanto cito algumas sob o risco de ser ingrata com muitas. Mas, sem dúvida, todas estarão presentes em minha lembrança.

Inicialmente, agradeço a Deus por todas as provisões que tornaram este dia uma realidade.

À Coordenação de Pós-Graduação de Engenharia Elétrica por ter-me concedido a oportunidade de ingressar no Programa de Mestrado.

Ao Prof. Edson, por ter-me dado autonomia para que encontrasse o meu caminho. Apesar de minhas “caminhadas em círculos”, continuava acreditando em mim, fazendo com que atalhos fossem tomados sempre que possível. Sua confiança e orientação foram estimulantes a cada momento difícil que atravessei.

Ao Departamento de Informática por ter me concedido o tempo necessário e permitir que desfrutasse de seus laboratórios.

Aos colegas e amigos da Pós-Graduação, cujo convívio foi um grande aprendizado. Em especial a Bruno Feres, Nilson Santos e Rômulo Alves, por terem colaborado significantemente para o desenvolvimento deste trabalho. A todos muito obrigada.

Aos meus familiares, em geral, que suportaram meus “altos e baixos” durante esses anos. Principalmente meu esposo e filhos que compartilharam comigo todos os momentos e ajudaram-me a vencer todas as minhas dificuldades.

Aos meus amigo(a)s que, apesar de estarem ocupados com seus trabalhos, sempre arrumavam um tempo para me motivar na realização desta dissertação.

A todos que, direta ou indiretamente, contribuiram para a elaboração desta dissertação.

RESUMO

Técnicas avançadas de detecção de intrusos em redes de computadores tornam-se cada vez mais importantes para prevenir abusos e proteger informações no ambiente. Esta dissertação apresenta uma proposta de um sistema de detecção de intrusos em redes de computadores, baseado na noção de sociedade de agentes inteligentes e redes neurais, denominado NIDIA. Uma implementação computacional é feita dos agentes sensores de rede e de *host* para realizar a tarefa de captura de pacotes associados às conexões suspeitas ou comportamentos anormais em servidores críticos.

Palavras Chaves: detecção de intrusos, segurança de redes de computadores, sistemas multiagentes, redes neurais.

ABSTRACT

Recently, the interest for advanced techniques for network intrusion detection have been increased for protecting important information in computational environment. This research work presents a proposal of a new network intrusion detection system based on a society of intelligent agents whose reasoning are aupported by neural network paradigms, named NIDIA (Network Intrusion Detection System based on Intelligent Agents). A computational implementation has been carried out for the network and host sensors for dealing with task of capturing packets related to suspicious connections or abnormal behaviors within critical hosts.

Keywords: network intrusion detection, network security. multi-agents systems, neural networks.

SUMÁRIO

LISTA DE ABREVIATURAS E SÍMBOLOS

AAFID	Agentes Autônomos para Detecção de Intrusão
AID	The Intrusion Detection System
ANS	Agente Servidor de Nomes
API	Aplication Program Interface
CERIAS	*Center of Education and Research in Information Assurance and Security at Purdue University*
CERT	Computer Emergency Responce Team
CFP	Chamada para proposta
UCP	Unidade Central de Processamento
CSI	Computer Security Institute
DF	Do Not Fragment
DFDB	base de dados de incidentes de intrusão
DLL	*dynamic linked library*
DoS	Denial of Service
E/S	Dispositivos de Entrada e Saída
EVT	*Event Viwer Type*
FDDI	Fiber Distributed Data Interface
FTP	Transferência de arquivos
GPF	General Protection Fault
HostAgent	Agente sensor de *host*
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
ICMP	*Internet Control Message Protocol*
ICMC/USP	Instituto de Ciências Matemáticas de São Carlos/Universidade de São Paulo
IDC	International Data Corporation
IIDB	Base de dados de padrões de intrusões
IP	Internet Protocol
IPA	Agência de Promoção de Tecnologia da Informação
IPV4	*Internet Protocol* Versão 4

IPV6	*Internet Protocol* Versão 6
IRC	*Internet Relay Chat*
JAM	Java Agents for Meta-Learning
LSIA	Agente de Segurança Local
MF	More Fragments
MIME	*Multipurpose Internet Mail Extensions*
MLSI	*Marks Left by Suspected Intruder*
NAT	*NetBIOS Auditing Tool*
NetAgent	agente sensor de rede
NFS	Network File System
NIDIA	*Network Intrusion Detection System based on Intelligent Agents*
NY	New York City
OE	Objeto de Eventos
ONG	Organização Não Governamental
OOB	Out Of Band
RABD	Base de dados de ações
SAA	Agente de Atualização do Sistema
SAI	Agente de Integridade do Sistema
SCA	Agente Controlador de Ações
SEA	Agente de Avaliação de Segurança do Sistema
SDI	Sistema de Detecção de Intrusos
SLIP	*Serial Line Internet Protocol*
SMA	Agente de Monitoramento de Sistema
SNMP	*Simple Network Management Protocol*
SNMP	Gerenciamento de redes
SO	Sistema Operacional
STBD	base de dados de estratégias
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UFMA	Universidade Federal do Maranhão
UML	Linguagem de Modelagem Unificada
VPN	Virtual Private Network

LISTA DE APÊNDICES

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Neste capítulo, faz-se a apresentação deste trabalho através de uma visão geral da problemática da segurança das informações, expondo a forte relação de dependência entre a sociedade atual e a informatização. Em seguida, abordam-se os objetivos gerais e específicos e a organização da dissertação.

1.1 Cenário atual

A tecnologia de Internet tendo experimentado um crescimento acelerado notadamente voltado para atividades essenciais e comerciais, tornou-se a base da chamada Nova Economia, onde a informação e o uso da tecnologia são fundamentais para o progresso das empresas.

Este novo panorama traz consigo muitos benefícios às organizações, tais como redução de custos e o fornecimento de serviços cada vez mais atraentes aos clientes, além de promover grandes oportunidades de negócios (*business-to-business* e *business-to-customers*).

Dessa forma, como a tecnologia da informação tem se tornado uma grande ferramenta estratégica (Módulo, 2000), a preocupação quanto à autenticidade, confidencialidade[1] e integridade[2] das informações passou a ser um fator crucial, uma vez que os prejuízos com invasões e incidentes de rede causam, a cada ano, grande impacto nas corporações (MÓDULO, 2001a).

De fato, tornou-se imprescindível a presença de mecanismos que visem dar apoio à segurança computacional e agilidade no processo de tomada de decisão, no caso de ausência do administrador de segurança, em decorrência de que a maioria das intrusões ocorre fora do expediente normal das corporações e um

[1] Confidencialidade é proteger informações confidenciais contra revelações não autorizadas ou captação compreensível.
[2] Integridade é manter informações e sistemas computadorizados, dentre outros ativos (elementos aos quais a organização atribui valor e desta forma requerem proteção), exatos e completos.

invasor pode permanecer durante semanas em um ambiente, estudando a melhor forma de efetuar o ataque[3], sem que seja identificado.

Um ataque pode ser ativo ou passivo, tendo por resultado a alteração ou liberação dos dados, respectivamente. O fato de um ataque estar acontecendo não significa necessariamente que ele terá sucesso. O nível de sucesso depende da vulnerabilidade[4] do sistema ou da atividade e eficácia de contramedidas existentes. Segundo o Computer Emergency Response Team (CERT/CC), os vírus Code Red e Nimda, que exploram vulnerabilidades conhecidas, foram os principais responsáveis pelas invasões ocorridas em 2001 (MODULO, 2002).

Este cenário mostra a necessidade do uso de Sistema de Detecção de Intrusos (SDI), para proporcionar maior grau de segurança às redes de computadores, mesmo àquelas que já possuam: *i)* uma política de segurança[5] bem definida; *ii)* uma equipe preocupada com os riscos[6], ameaças[7] e as correções de vulnerabilidades dos servidores e redes; *iii)* várias camadas de proteção, composta por várias tecnologias trabalhando em conjunto, como parte de uma solução geral de segurança (p. ex. anti-vírus, *firewalls*, autenticação, smart cards, criptografia, VPN).

Atualmente, vistos como uma solução inovadora (Módulo, 2001), os SDI´s estão sendo largamente utilizados nas corporações, instituições governamentais e redes de computadores acadêmicas, como uma poderosa ferramenta de auxílio aos administradores de segurança.

Segundo o IDC (*International Data Corporation*), o mercado das ferramentas SDI atingiu uma venda de 136 milhões de dólares em 1998. Em 1999, foi verificado um crescimento de aproximadamente 100%, e estima-se que até 2003

---

[3] Define-se ataque como sendo tentativa, frustrada ou não, de entrar em locais não autorizados.
[4] Vulnerabilidades são pontos fracos associados a um ativo (por ex. servidor crítico) ou grupo de ativos, os quais podem ser explorados por um atacante.
[5] Política de segurança é o documento formal que define, através de procedimentos e normas, quais atividades são permitidas na rede das organizações ou em servidores particulares.
[6] Risco é a possibilidade de uma dada ameaça explorar vulnerabilidades de um ativo ou grupo de ativos para causar perdas e danos a estes.
[7] Ameaças são causas potenciais de incidentes não esperados, que podem resultar em danos aos ativos da organização (Ex.: acesso e uso não autorizado de informações, roubo, ataques que exploram o acesso confiável, DoS, Vírus etc).

atingirá um mercado de 980 milhões de dólares (PRNEWSWIRE, 1999). Portanto, o mercado de ferramentas de detecção de intrusos está, a cada ano, atraindo mais investidores, tornando-se, dessa forma, uma área bastante competitiva e movimentando milhões de dólares.

1.2 Definição do problema

Atualmente, a presença de empresas brasileiras ligadas à Internet é bastante expressiva[8], sendo que, conforme recente pesquisa realizada pela Módulo Security Solutions, que entrevistou 165 executivos de grandes empresas públicas e privadas distribuídas entre os segmentos financeiro, serviços, informática, indústria, telecomunicações, governo, e-commerce e varejo, foi verificado que:

- somente em 3% das empresas não é permitido o acesso à Internet;
- 72% disponibilizam acesso à Internet a seus funcionários através de sua rede interna;
- 16% permitem o acesso via modem nas empresas e 19% autorizam o acesso remoto via modem na residência.

Outro dado importante é que 75% das empresas brasileiras possuem página na Web, sendo que:

- cerca de 63% desse total possuem aplicações de *Internet Banking* (10%), vendas (6%) e consultas a bancos de dados (6%);
- dentre as empresas que afirmaram realizar transações eletrônicas via Internet (proximadamente 59%), destacam-se os serviços de vendas (25%), bancários (14%) e compras (7%).

O estudo também aponta para um crescimento de 57% nos serviços de *e-commerce* a partir de 2001 (MÓDULO, 2001a).

[8] O Brasil ocupa o 11º lugar em número de hosts do mundo e o 1º lugar na América do Sul, segundo indicadores fornecidos pelo Comitê Gestor da Internet do Brasil em janeiro de 2002 (CG, 2002).

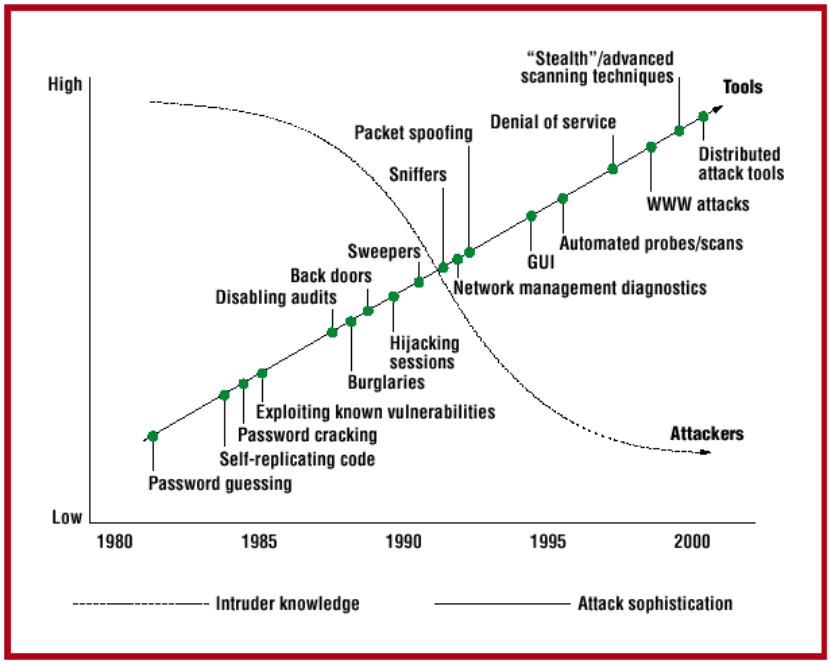
Contudo, o número de incidentes registrados aumentou proporcionalmente ao crescimento da Internet. De acordo com o CERT (*Computer Emergency Responce Team*), de 1998 a dezembro de 2001, foram registrados 100.369 incidentes de segurança (tentativas de invasão com ou sem sucesso), sendo que 21.756 ocorreram em 2000 e 52.658 só em 2001 (CERT, 2002). Esses números alarmantes, que são ainda maiores, visto que muitas corporações não divulgam ou desconhecem o fato de terem sido invadidas[9], estão associados a prejuízos de milhões de dólares.

Em uma pesquisa realizada nos Estados Unidos, o CSI (*Computer Security Institute*) entrevistou 538 profissionais de segurança da informação em corporações norte-americanas, agências de governo, instituições financeiras, instituições médicas e universidades, revelando que as empresas que conseguiram quantificar suas perdas financeiras, no total de 35%, contabilizaram prejuízos de 377 milhões de dólares, sendo 112 milhões a mais do que em 2000 (CSI, 2001).

No Brasil, segundo a Módulo Security Solutions, das empresas que tiveram perdas causadas por invasões, somente 15% puderam especificar em valores seus prejuízos. Nesse contexto, 88% admitiram ter perdas menores que R$ 50 mil, 4% de R$ 50 mil a R$ 500 mil e 8% estimaram entre R$ 500 mil e R$ 1 milhão[10] suas perdas financeiras com crimes virtuais (MÓDULO, 2001a).

Este cenário pode ser explicado pelo fato de que cada vez mais as técnicas de invasão estão acessíveis para os usuários, permitindo que agressores, com conhecimentos técnicos limitados, mas dispondo de ferramentas sofisticadas, possam efetivar ataques bem sucedidos, conforme mostrado na Figura 1.1.

[9] Segundo pesquisa recente, cerca de 31% das empresas brasileiras entrevistadas desconhecem o fato de terem sido invadidas (MÓDULO, 2001a).
[10] Este índice manteve-se inalterado desde 2000.



Fonte: (CERT, 2000a).

Figura 1.1 - Sofisticação de Ataques x Conhecimento Técnicos dos Atacantes.

Ao contrário do que ocorre hoje, nos anos 80 os intrusos eram especialistas, com alto nível de conhecimento, que desenvolviam seus próprios métodos de ataques a um sistema. Eles raramente utilizavam ferramentas automatizadas e exploravam *scripts*. Até meados de 1990, os ataques de negação de serviço (DoS – *Denial of Service*), tidos como os grandes vilões dos dias de hoje, capazes de comprometer a credibilidade ou até mesmo afastar empresas de seus ramos de negócios, não eram freqüentes e não representavam grandes riscos às corporações.

No Brasil, os grupos de *hackers*[11] têm aumentado consideravelmente. Segundo o *site* alldas.de, que registra invasões na Internet, as equipes *Silver Lords* e *Prime Suspectz* apareceram, respectivamente, em primeiro e terceiro lugar no *ranking* dos *hackers* mais perigosos de 2001 (ALLDAS, 2001).

[11] Pessoas que tentam acessar sistemas sem autorização, usando técnicas próprias ou não, no intuito de ter acesso a determinado ambiente para proveito próprio ou de terceiros. Dependendo dos objetivos da ação, podem ser chamados de Cracker, Lammer ou BlackHat.

Com tudo isso, a preocupação com a segurança da informação ganhou lugar estratégico na política das organizações, sendo que, algumas medidas de segurança estão sendo largamente utilizadas, destacando-se, no Brasil, o firewall (83%), a prevenção contra vírus (78%), o proxy server (71%), a segurança na sala de servidores (62%) e o backup (61%). A pesquisa destaca o fato de que os sistemas de detecção de intrusos aparecem, pela primeira vez, como medida de segurança adotada em 37% das empresas entrevistadas (MÓDULO, 2001a).

Entretanto, ainda que os firewalls (Ranum, 1992) sejam considerados uma alternativa segura e largamente utilizada, seu uso não implica segurança total, uma vez que o mesmo não protege contra códigos móveis maliciosos[12] (applets Java e objetos ActiveX), acesso discado[13] e o seu uso torna-se difícil em redes distribuídas, sem um ponto central de tráfego (CERT, 2000).

Constantemente são divulgadas notícias de vulnerabilidades descobertas nesses sistemas (SFOCUS, 2001; ISS, 2001). Além disso, qualquer erro ou esquecimento na sua configuração é o suficiente para que se caracterize uma falha na segurança que poderá ser explorada[14]. Outro problema é em relação aos maus usuários (usuários legítimos com má intenção), que continuam sendo grandes responsáveis pelo número de incidentes ocorridos nas corporações[15] (MÓDULO, 2000, 2001a; CSI, 2000).

Portanto, se faz necessário o uso de outras tecnologias atuando em conjunto para aumentar o nível de segurança de uma rede privada, visto que, isoladamente, nenhuma oferece um elevado grau de segurança, permitindo simultaneamente uma certa flexibilidade e liberdade no uso dos recursos protegidos.

---

[12] Alguns firewalls podem prevenir a importação de código Java e ActiveX através do reconhecimento de tipos MIME. Porém, essa funcionalidade não fornece a habilidade de se distinguir entre um código legítimo e um malicioso e, com isso, tem que ser configurado para aceitar ou não todo tipo de código ActiveX ou Java.

[13] Segundo pesquisa realizada pela Módulo Security Solutions, no Brasil, 38% das empresas que usam o firewall como método de defesa, permite o acesso à Internet via modem.

[14] Segundo pesquisa realizada pela empresa britânica NTA Monitor, 50% das vulnerabilidades encontradas em firewalls são ocasionadas pela má configuração feita pelos administradores de sistemas. O estudo revela que de 1998 até 2002, houve um aumento de 45% no número de falhas (MODULO, 2002a).

[15] Pesquisas recentes realizadas pela Módulo Security Solutions e o Computer Security Institute Press Release revelam que as ações intrusivas oriundas da rede interna das organizações podem chegar a 70% do total.

1.3 Objetivo geral e específico

Esta dissertação objetiva propor um modelo de sistema de detecção de intrusos, baseado na noção de sociedade de agentes inteligentes, capaz de detectar e tratar as tentativas de ataque em uma rede de computadores de forma bastante flexível, denominado NIDIA (*Network Intrusion Detection System based on Intelligent Agents*). O Projeto NIDIA tem o objetivo de fornecer uma contribuição para a melhoria das técnicas de detecção de intrusos em redes de computadores, permitindo que vários pesquisadores, incluindo-se professores, alunos da graduação e pós-graduação, tenham a oportunidade de conhecer, trabalhar e disseminar as novas tecnologias das áreas de segurança de redes de computadores, inteligência artificial e sistemas multiagentes.

Este trabalho tem como objetivos específicos:

a) apresentar o modelo para detecção de intrusos, definindo a funcionalidade básica de cada agente e a interação em sociedade, identificando-se os principais Casos de Uso, através da Linguagem de Modelagem Unificada (UML) (BOOCH et al., 1999);

b) apresentar a implementação do agente sensor de rede, responsável pela captura de pacotes, a partir do tráfego da rede;

c) apresentar a implementação do agente sensor de *host*, responsável pelo monitoramento de eventos ocorridos em máquinas Windows NT 4.0 e Windows 2000;

d) demonstrar a viabilidade do projeto, através da implementação do protótipo dos agentes sensores, enfatizando-se a situação atual e os futuros trabalhos a serem desenvolvidos.

1.4 Organização do Trabalho

Esta dissertação encontra-se organizada em seis capítulos. No Primeiro Capítulo é apresentado o cenário atual do mercado das ferramentas de segurança, especificamente os Sistemas de Detecção de Intrusos, o crescimento progressivo do uso da Internet voltados para atividades comerciais, informações técnicas quanto ao aumento de incidentes registrados nos últimos anos, bem como os objetivos gerais e específicos e a organização da dissertação.

No Capítulo 2, apresenta-se uma visão geral dos sistemas de detecção de intrusos, discutindo-se conceitos básicos, características e aplicações, enfatizando-se as vantagens dos sistemas de detecção de intrusos baseados na tecnologia de agentes e os trabalhos relacionados nessa área.

No Capítulo 3 é proposto um modelo de detecção de intrusos em rede de computadores, denominado NIDIA, baseado na tecnologia de agentes e apresentando-se a sua arquitetura e funcionalidade através de Casos de Uso da notação UML.

O Capítulo 4 apresenta a operacionalização do modelo proposto, destacando-se a implementação do protótipo dos agentes sensores de rede e de host.

No Capítulo 5 mostra-se os resultados obtidos das simulações realizadas com o protótipo trabalhando em sociedade, através de eventos capturados e analisados.

O Capítulo 6 apresenta as conclusões finais, enfatizando as contribuições desta dissertação e algumas indicações para trabalhos futuros.

2 SISTEMAS MULTIAGENTES PARA A DETECÇÃO DE INTRUSOS

Apresenta-se, neste capítulo, uma visão geral sobre os Sistemas de Detecção de Intrusos (SDI) como ferramenta de auxílio na administração de segurança, apresentando suas vantagens, limitações, características, seus métodos de análise, classificação, tempo de coleta e resposta aos incidentes. Por fim, mostram-se as vantagens dos sistemas de detecção de intrusos baseados na tecnologia de agentes sobre os sistemas monolíticos, bem como os trabalhos relacionados nessa área.

2.1 Considerações gerais

A Inteligência Artificial (IA) é o ramo da ciência que estuda o fenômeno da inteligência, com o objetivo de modelar e simular comportamentos inteligentes em máquinas para resolver problemas complexos sem ter, necessariamente, o auxílio humano. Para tanto, vários processos de raciocínio, aprendizado e percepção são necessários.

Nas abordagens clássicas de IA, a ênfase da inteligência é baseada em um comportamento humano individual e o foco de atenção volta-se à representação de conhecimento e métodos de inferência. Já a Inteligência Artificial Distribuída (IAD) baseia-se no comportamento social e sua ênfase é nas cooperações, nas interações e no fluxo de conhecimento entre unidades distintas. Na resolução distribuída de problemas, essas entidades, conhecidas como agentes, cooperam umas com as outras, dividindo e compartilhando conhecimentos sobre questões propostas e processos de obtenção de soluções. Nessa abordagem, os agentes são projetados para atuarem em domínios específicos que correspondem a subconjuntos do problema principal. A solução do problema é feita com a ajuda de todos os agentes envolvidos de maneira não conflitante (GASSER; HUHNS, 1989; HOGG; HUBERMAN, 1991; OLIVEIRA, 1996).

Um ambiente multiagentes pode ser definido como um sistema no qual diversos agentes (sociedade de agentes) interagem para solucionar problemas (LIZOTTE; MOULIN, 1990).

Esta concepção de sociedade de agentes cooperativos tem influenciado várias áreas de pesquisa. Uma das aplicações possíveis no campo de segurança de ambientes informatizados é o uso de agentes na detecção de intrusos.

No contexto deste trabalho, os agentes são modelados como entidades computacionais que podem realizar funções de monitoramento, detecção e resposta às atividades maliciosas em uma rede de computadores e servidores críticos.

2.2 Conceitos básicos

Segundo (Crosbie e Spafford, 1995a), uma intrusão pode ser definida como sendo "*um conjunto de ações que tentam comprometer a integridade, confidencialidade ou disponibilidade*[16] *de recursos*" em um sistema computacional.

Um sistema de detecção de intrusos (SDI) realiza o processo de monitorar os eventos[17] ocorridos em um sistema ou rede de computadores, analisando-os através de assinaturas de intrusão ou desvio de padrão em perfis de comportamento. As intrusões são causadas por usuários[18] não autorizados tentando acessar um sistema através da Internet e por usuários legítimos tentando abusar de seus privilégios, motivados por: ganhos financeiros; vingança; necessidade de aceitação ou respeito; idealismo; curiosidade ou busca de emoção; aprendizado ou espionagem industrial.

[16] Disponibilidade é garantir que informações e serviços vitais estejam disponíveis quando requeridos.
[17] Um evento é dito como sendo a ocorrência que um SDI usa para detectar uma atividade indesejada ou desautorizada, que poderá resultar em um alerta. Por exemplo, "N" falhas de logon em "T" segundos pode indicar um ataque de "*brute-force login"*.
[18] Usuário, no contexto do trabalho, tem-se como um servidor, uma estação de trabalho, um endereço ou faixa de endereços IP, um nome de domínio, ou simplesmente um indivíduo.

O conceito de sistema de detecção de intrusão foi proposto pela primeira vez em 1980 por James Anderson (Anderson, 1980), sendo difundida somente em 1987, com a publicação do artigo “Um Modelo de Detecção de Intrusão” por Dorothy Denning (DENNING, 1987). Em 1988, pelo menos três propostas de SDI foram apresentadas: NIDX - Um Sistema Especialista para Detecção de Intrusos em Redes de Computadores em Tempo Real (Bauer e Koblentz, 1988); Sistemas Especialistas em Detecção de Intrusos: Um Estudo de Caso (Sebring et al., 1988) e *Haystack*: Um Sistema de Detecção de Intrusão (SMAHA, 1988). Nos anos seguintes, vários protótipos foram desenvolvidos.

2.3 Vantagens dos sistemas de detecção de intrusão

Os sistemas de detecção de intrusos possuem diversas características que lhes conferem papel importante dentro da infra-estrutura de segurança de uma organização. Em (Crosbie e Spafford, 1995; Zamboni et al, 1998), as seguintes características são identificadas como desejáveis para esses sistemas:

- estar em execução contínua com a mínima supervisão humana;
- ser tolerante a falhas. O sistema deve ter a habilidade de recuperar-se após qualquer interrupção, seja acidental ou por atividades maliciosas e retornar a atividade normal de funcionamento após ser reiniciado;
- resistir à subversão. Os sistemas de detecção de intrusos devem ter a habilidade de monitorar-se contra invasões, de forma a não se tornar um veículo de ataque;
- gerar o mínimo de *overhead* possível no ambiente onde está sendo executado, de forma a não interferir nas operações normais;
- ter a habilidade de ser configurado de acordo com a política de segurança do ambiente que está sendo monitorado;

- adaptar-se às mudanças do ambiente e dos comportamentos dos usuários através do tempo, uma vez que novas aplicações sendo instaladas, usuários trocando de atividade ou novos recursos sendo disponibilizados podem causar mudanças nos padrões de uso dos recursos do ambiente;

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

- apresentar uma moderada degradação de serviço, no caso de algum componente do sistema ter seus serviços interrompidos por qualquer motivo;

- permitir reconfiguração dinâmica, ou seja, não ser necessário reiniciar o SDI toda vez que um novo cenário for implementado.

As vantagens mais comuns dos SDI´s, segundo (Bace e Mell, 2001), são relatadas a seguir:

a) Detecção de ataques explorando vulnerabilidades que não podem ser corrigidas

Através de vulnerabilidades conhecidas, que em alguns casos não podem ser corrigidas, o invasor pode penetrar em muitas redes. A exemplo disso, têm-se os sistemas herdados, onde o sistema operacional não pode ser atualizado.

Mesmo em sistemas passíveis de atualização, os administradores podem não dispor de tempo para instalar todos os *patches* necessários em uma grande quantidade de computadores. Outra situação é quando o próprio negócio requer serviços e protocolos de rede que são notoriamente problemáticos e sujeitos a ataques. Embora, de maneira ideal, todas essas vulnerabilidades possam ser sanadas, na prática isso nem sempre é possível. Assim, uma boa solução pode ser o uso de SDI para detectar e, possivelmente, prevenir tentativas de ataque explorando estas vulnerabilidades.

b) Prevenir que intrusos examinem a rede.

Quando os intrusos atacam uma rede, eles geralmente o fazem em etapas. A primeira etapa é examinar o sistema em busca de alguma vulnerabilidade. Na ausência de um sistema de detecção de intrusão, o atacante se sente livre para fazer o seu trabalho sem riscos e pode descobrir com facilidade os pontos fracos do sistema e explorá-los. Com um SDI, embora o iminente intruso possa continuar a analisar o sistema, ele será detectado e os devidos alertas[19] serão dados.

c) Documentação de ameaça de invasão.

É importante entender a freqüência e as características dos ataques em um sistema computacional para medir o grau de hostilidade de um ambiente. Com isso, é possível tomar as medidas de segurança adequadas para proteger a rede. Um SDI pode quantificar, caracterizar e verificar as ameaças de intrusão e mau uso, interna ou externamente.

d) Controle de qualidade

Com o uso contínuo de um SDI, padrões de utilização do sistema monitorado e problemas detectados tornam-se evidentes. Com essas informações, é possível tornar visíveis, falhas no projeto, configuração e gerenciamento da segurança, proporcionado ao administrador a possibilidade de corrigi-las antes de ocorrer um incidente.

Essas vantagens sofrem variações de acordo com alguns aspectos básicos que devem ser considerados quando do desenvolvimento dos SDI´s, tais como: os métodos de análises adotados, a escolha do tipo de monitoramento, o tempo entre a coleta e análise dos dados e o tipo de resposta a ser dada. Esses tópicos serão vistos com mais clareza nas seções seguintes.

[19] Alerta é dito como sendo uma mensagem de que um evento foi detectado. Contém tipicamente informação sobre a atividade que foi detectada, bem como as especificações das ocorrências (origem, destino, nome, tempo do evento, porta, serviço, informação de usuários e outros).

2.4 Métodos de análise de detecção de intrusão

Existem dois métodos principais de análise de detecção de intrusão, usados pelos SDI´s, para determinar a ocorrência de uma intrusão, conforme descritas em (Bace e Mell, 2001): **a detecção por anomalia e a detecção por abuso**.

2.4.1 Detecção por anomalia

O método de detecção de intrusos por anomalia é baseado na observação de desvios de comportamentos esperados em atividades[20] relevantes dos usuários ou processos do sistema monitorado. Os detectores de anomalia constroem perfis (*profiles*) que representam o comportamento normal dos usuários, *hosts* ou conexões de rede. Esses perfis são gerados através de dados históricos coletados durante um período normal de operação. Os sistemas de detecção de intrusos por anomalia baseiam-se na premissa de que um ataque difere da atividade normal, podendo, dessa forma, ser identificado.

Porém, nem sempre a atividade anômala corresponde a uma atividade intrusiva. Por exemplo, um usuário que sempre trabalhou com determinada carga de uso da UCP pode necessitar de mais poder de processamento e nem por isso ele é um intruso. Em (KUMAR, 1995), são apresentadas as seguintes definições de comportamento:

- **intrusivo e anômalo**: a intrusão coincide com a anormalidade. São os verdadeiros positivos;
- **não intrusivo e não anômalo**: não há intrusão, nem anormalidade. São verdadeiros negativos.

[20] Ocorrências detectadas pelos sensores como sendo de interesse do administrador., tais como: uma sessão de telnet inesperada, usuário tentando alterar objetos sem ter privilégios ou arquivos de log mostrando persistentes falhas de logon.

- **intrusivo mas não anômalo**: há intrusão sem que haja comportamento anormal. São os falsos negativos.

- **não intrusivo mas anômalo**: não há intrusão, mas há comportamento anormal. São os falsos positivos.

A ocorrência de um grande número de falsos negativos e falsos positivos pode se tornar um problema dos SDI´s baseados em detecção por anomalia. Para reduzi-los, é preciso a definição cuidadosa dos limites que apontam a anormalidade.

Para sustentar a implementação dos SDI´s baseados nesse tipo de detecção, vários métodos foram propostos. As abordagens mais comuns são: o método estatístico, o gerador de prognósticos de padrões e o uso de redes neurais.

No **método estatístico** (Lunt et al, 1992), inicialmente, os perfis de comportamento dos usuários são criados. Isto é feito através de um período de treinamento onde os usuários são monitorados em suas atividades cotidianas. O comportamento padrão é, então, mantido num perfil particular ou mesmo em perfis de grupos. À medida que o sistema continua a operar normalmente, o detector cria o perfil atual, em intervalos regulares e compara-lhe com o perfil original armazenado. Assim, se o desvio for demasiadamente grande, pode-se inferir que há sinais de anormalidade, o que poderia implicar intrusão.

Um assunto ainda em aberto com o modelo estatístico na detecção por anomalia é a escolha do que deve ser monitorado. As métricas que afetam o perfil de comportamento podem ser as mais diversas, como: quantidade de arquivos abertos, tempo de uso da UCP, número de conexões de rede em um período, operações de E/S, etc. Não é conhecido, de fato, qual subconjunto de todas as medidas possíveis, consegue predizer melhor atividades intrusivas. Portanto, uma boa análise do conjunto de medidas utilizadas contribui bastante para o sucesso do método estatístico.

As principais vantagens desse método são que ele pode ser baseado em técnicas estatísticas bem conhecidas e aprender o comportamento dos usuários, adequando-se a mudanças no perfil.

É justamente devido a essa grande adaptabilidade que ocorre a principal fragilidade dessa abordagem, já que os sistemas baseados em estatística podem gradualmente ser treinados por intrusos de tal forma que eventualmente os eventos intrusivos sejam considerados como normais.

Outra desvantagem desse método é a dificuldade em se determinar um grau de tolerância adequado para que um perfil não seja considerado intrusivo (*threshold*). Se ele for muito alto ou muito baixo, então poderá haver um grande número de falso negativo ou falso positivo.

Alguns problemas associados a esse método foram remediados por outras abordagens, incluindo o método conhecido como gerador de prognósticos de padrões, que leva em consideração a ordem dos eventos passados em sua análise.

No **gerador de prognósticos de padrões**, tenta-se predizer os eventos futuros com base em eventos que já ocorreram, conforme expõe (TENG; CHENG, 1990). Portanto, pode-se ter uma regra do tipo: E1 - E2 → (E3 = 80%, E4 = 15%, E5 = 5%). Isto significa que dados os eventos E1 e E2, com E2 ocorrendo depois de E1, existe 80% de chance de que o evento E3 ocorra em seguida, 15% de chance que E4 seja o próximo e 5% que E5 ocorra. Assim, na ocorrência de um determinado evento, ou uma série de eventos, é possível saber quais os possíveis eventos subseqüentes.

Da mesma forma que no método anterior, aqui é necessário primeiro haver um período de treinamento. Através do acompanhamento dos usuários em suas atividades diárias, regras temporais que caracterizam os padrões de comportamento podem ser geradas. Essas regras se modificam ao longo do período de aprendizado. Porém, somente são mantidas no sistema aquelas que apresentarem um desempenho ótimo no prognóstico, ou seja, só as regras que conseguem predizer o próximo evento de forma correta são aceitas.

Há algumas vantagens nessa abordagem:

- padrões seqüenciais baseados em regras podem detectar atividades anômalas que são difíceis nos outros métodos;

- sistemas construídos usando esse modelo são bastante adaptativos a mudanças. Isto se deve ao fato de que padrões de má qualidade são continuamente eliminados, conservando-se os padrões de alta qualidade;

- é mais fácil detectar usuários que tentam treinar o sistema durante o período de aprendizagem;

- atividades anômalas podem ser detectadas muito rapidamente a partir do recebimento dos eventos de auditoria.

Um problema desse método é que determinados padrões de comportamento podem não ser detectados como anômalos, simplesmente porque não é possível combiná-los em nenhuma regra. Essa deficiência pode ser parcialmente resolvida acusando-se todos os eventos desconhecidos como intrusivos, correndo-se o risco de aumentar o número de falsos positivos ou como não intrusivos, aumentando-se a probabilidade de falsos negativos. No caso normal, entretanto, um evento é acusado como intrusivo se o lado esquerdo da regra é igualado, mas o lado direito é estatisticamente muito diferente da predição.

Como uma possível solução para esse problema, o uso de **redes neurais** tem sido pesquisado (FOX et al., 1990). A idéia básica é o treinamento de uma rede neural para predizer a próxima ação do usuário, dado o conjunto de suas ações anteriores. A rede é treinada com o conjunto das ações representativas.

Qualquer evento predito que não corresponda com a realidade, pode medir um desvio no perfil normal do usuário. Algumas vantagens dessa abordagem são que as redes neurais têm a capacidade de inferir dados com ruídos; o seu sucesso não depende de nenhuma suposição sobre a natureza dos dados e são mais fáceis de se modificar para uma nova comunidade de usuários.

Entretanto, esse método tem alguns problemas, pois um pequeno conjunto de informações irá resultar em muitos falsos positivos, enquanto que um conjunto grande com informações irrelevantes irá aumentar as chances de falsos negativos. Além disso, a topologia da rede só é determinada depois de muita tentativa e erro.

Resumindo-se, o ponto forte desse tipo de detecção é a capacidade de reconhecer tipos de ataques que não foram previamente cadastrados, baseado apenas nos desvios de comportamento, enquanto que os pontos fracos podem ser o grande número de alarmes falsos gerados para uma rede treinada insuficientemente e a necessidade de um grande período de treinamento *off-line* para se construir os perfis dos usuários.

2.4.2 Detecção por abuso

A detecção de intrusos por abuso está relacionada à identificação de comportamentos intrusivos correspondentes a exploração de vulnerabilidades conhecidas, comumente chamadas de assinaturas de ataque[21]. Os detectores de abuso analisam a atividade do sistema, observando os eventos ou conjunto de eventos que sejam semelhantes a um padrão pré-determinado que descreva uma intrusão conhecida.

As assinaturas não servem apenas para detectar intrusões consumadas, elas são úteis, também, para identificar tentativas de ataque. Assim, quando um conjunto de eventos satisfaz apenas parcialmente uma assinatura, pode ser um indicativo de uma intrusão futura que já começou.

São conhecidas várias abordagens para implementar SDI´s baseados em detecção por abuso. As mais comuns são: a utilização de sistemas especialistas; a detecção por modelo; a análise de estados de transição e o uso de redes neurais.

[21] Análise de assinatura é a busca por padrões de configuração de sistema ou de atividades do usuário em um banco de dados de ataques conhecidos. Portanto, os SDI´s devem ser programados para detectar cada tipo conhecido de ataque.

Os **Sistemas especialistas** são sistemas inteligentes que capturam o conhecimento de especialistas humanos em domínios limitados, geralmente na forma de regras do tipo IF-THEN. A combinação entre as fases de aquisição dos dados e a ação propriamente dita é feita de acordo com a comparação entre os eventos atuais, capturados através dos mecanismos de auditoria do sistema operacional e as regras já estabelecidas pelo engenheiro de conhecimento. As condições "IF" do sistema especialista codificam o conhecimento sobre os ataques, declarando suas características. Então, satisfeitas tais condições, as ações "THEN" são executadas (SNAPP; SMAHA, 1992).

Há algumas vantagens nessa abordagem, a saber: *i)* com um conjunto de regras bem definido, é possível ter-se um sistema ágil e com pouca sobrecarga; *ii)* vários critérios podem ser analisados para se identificar um ataque e ainda manter o sistema especialista leve e eficiente; e, teoricamente, é possível deduzir simbolicamente a ocorrência de uma intrusão baseada apenas nos dados recebidos.

O método apresenta diversas desvantagens. Somente ataques conhecidos e bem definidos podem ser detectados, o que exige uma constante atualização da base de regras. Segundo (Lunt, 1993), os sistemas especialistas são criados à semelhança de quem os programou, sendo limitado pelo conhecimento do programador ou do responsável pela segurança. Outra desvantagem é a dificuldade que há em tirar conclusões sobre situações consideradas ambíguas, devido a existência de conflitos nos fatos assumidos pelos antecedentes das regras.

De acordo com a **Detecção por Modelo**, certos cenários[22] de ataque podem ser inferidos através do monitoramento de determinadas atividades. Assim, se um conjunto de eventos ocorrer da forma descrita nos cenários, então é possível detectar a tentativa de ataque. Esse modelo é baseado em três importantes módulos, conforme descrito por (GARVEY; LUNT, 1991).

[22] Cenários são seqüências de comportamentos que caracterizam uma intrusão.

O ANTICIPATOR busca no banco de dados de cenários aquele que melhor expressa as atividades atuais do sistema. Então, ele tenta predizer os próximos passos da possível tentativa de ataque de acordo com o cenário escolhido. Depois, essas informações são transmitidas ao PLANNER, encarregado de traduzir os passos hipotéticos em um formato compatível com sua provável representação no sistema de auditoria.

Por fim, ao INTERPRETER cabe procurar nos registros auditados somente os prováveis eventos. Com esse esquema, é possível supor as ações futuras que o invasor irá realizar e confirmar ou rejeitar tais suspeitas através do monitoramento e análise dos registros de *log* do sistema.

A vantagem dos SDI´s baseados no método de detecção por modelo é permitir a emissão de conclusões, ainda que parciais, sobre situações ambíguas. É também possível reduzir a quantidade de processamento exigida por cada registro auditor, através de uma monitoração superficial, buscando somente aqueles eventos necessários para a confirmação do cenário de intrusão.

A desvantagem dos SDI´s baseados nesse método é de não detectar situações que não tenham sido especificamente evidenciadas. Poucos são os relatos de protótipo desse modelo implementado, ficando-se dúvidas sobre a eficiência de sua execução, em função dos cálculos envolvidos.

Segundo (Porras e Kemmerer, 1992), a detecção por abuso baseada na análise de estado de transição considera que um sistema monitorado pode ser representado como uma seqüência de transição de estados. Uma transição acontece quando alguma condição booleana é tida como verdadeira (por exemplo, a abertura de um determinado arquivo). Sucessivos estados estão conectados por arcos que representam os eventos necessários para a mudança de estado ocorrer.

Os tipos de eventos permitidos são descritos no modelo e não precisam ter uma correspondência um a um com os registros de auditoria. Para o estado final (comprometido) ser alcançado, diversas condições precisam ser satisfeitas. Assim, se todas as condições intermediárias forem verdadeiras, então é quase certo que

uma tentativa de intrusão está ocorrendo. Entretanto, se alguma dessas condições é falsa, a probabilidade de uma intrusão diminui.

Algumas vantagens dessa abordagem é poder detectar ataques cooperativos, mesmo aqueles que variam entre várias sessões de usuário e prever o acontecimento de ataques iminentes, podendo-se, com isso, tomar medidas preventivas.

As desvantagens são que: *i)* os padrões de ataques podem especificar somente uma seqüência de eventos, ao invés de formas mais complexas; *ii)* não há um mecanismo eficiente que reconheça as tentativas parciais de ataque; *iii)* essa abordagem não pode detectar "negação de serviço", falhas de *login*, variações do uso normal etc, pois esses itens não são gravados pelos mecanismos de *log* e, portanto, não podem ser representados por diagramas de transição de estado.

Assim como na detecção de intrusos por anomalia, as **redes neurais** têm sido utilizadas para reconhecer padrões de ataques ou assinaturas. (BONIFÁCIO Jr. et al, 1998). Mas, diferentemente da primeira abordagem, aqui o mais interessante dessa tecnologia é sua capacidade de classificação e generalização.

Por exemplo, dado um ataque que não seja exatamente igual à sua descrição armazenada, a rede neural pode reconhecê-lo simplesmente pelas semelhanças existentes com a descrição. Cansian *et al* (1997) desenvolveram um sistema que utiliza redes neurais para detectar padrões de intrusão em redes de computadores. Ele atua capturando e decifrando pacotes, para realizar inferências acerca do estado de segurança das sessões, utilizando um sistema de pré-filtragem e classificação. Como resultado, a rede neural fornece um número que, baseado em informações de assinaturas de ataques anteriores, indicará a gravidade de um determinado evento.

As vantagens e desvantagens do uso de redes neurais na detecção por abuso são análogas àquelas apresentadas na detecção por anomalia.

De forma geral, a limitação principal da abordagem de detecção por abuso é que somente vulnerabilidades conhecidas são detectadas. Assim, novas

técnicas de intrusão devem ser constantemente adicionadas. Outra limitação desse método tem a ver com considerações práticas sobre o que é auditado.

Por outro lado, o método de detecção por abuso tem sido o mais utilizado pelos sistemas de detecção de intrusos, pois os estudos mostram que cerca de 93% das tentativas de intrusão ocorrem a partir de pequenas variações de padrões bem definidos de comportamento (CERT, 2000b). Além disso, outros fatores importantes que fazem com que esse método seja o mais preferido pelos sistemas comerciais, estão relacionados com à facilidade de configuração, custo computacional reduzido e pequeno comprometimento do desempenho. Quanto ao número de falsos positivos, a literatura disponível mostra que os sistemas de detecção de intrusos por abuso geram menos alarmes falsos do que os que usam o método de detecção por anomalia (SFOCUS, 2001a).

Conforme descrito anteriormente, os dois métodos possuem vantagens e desvantagens quanto à sua utilização. Conseqüentemente, uma combinação de ambos pode gerar um sistema de detecção de intrusos com maior capacidade para detectar atividades maliciosas em um ambiente computacional.

2.5 Categorias de sistemas de detecção de intrusão

Embora muitos sistemas de detecção sejam vistos conceitualmente como compostos por um sensor, um analisador e uma interface de usuário, os tipos de dados examinados e gerados pelos SDI´s podem variar significantemente. Os sistemas de detecção de intrusos podem ser classificados em uma das seguintes categorias (CERT, 2000), de acordo com os tipos de dados que eles examinam:

- **SDI baseado em rede**: a maioria dos sistemas de detecção de intrusos está classificada nessa categoria. Esses sistemas detectam intrusos através da captura e análise de pacotes que trafegam na rede, podendo, com isso, monitorar vários *hosts* conectados ao mesmo segmento de rede.

- **SDI baseado em host**: examina trilhas de auditoria e arquivos de *logs* para monitorar eventos locais em um sistema de computador em particular, podendo detectar ataques que não são vistos pelo SDI baseado em rede.

- **SDI baseado em aplicação**: examina o comportamento de um programa de aplicação, geralmente na forma de arquivos de *log*. Esses SDI´s são um subconjunto dos SDI´s baseados em *host*.

As vantagens dos SDI´s baseados em rede são que: *i)* se localizados em pontos estratégicos, podem monitorar uma rede bem extensa; *ii)* tem pouco impacto na rede monitorada, exigindo um mínimo esforço de instalação, pois os sensores são dispositivos passivos; *iii)* diversas técnicas podem ser usadas para camuflar os sensores de rede, deixando-os invisíveis aos intrusos; *iv)* é possível monitorar ataques distribuídos à rede.

Entretanto, essa categoria de SDI tem alguns pontos negativos, mostrados a seguir:

- em longos períodos de tráfego intenso na rede, os sensores podem não conseguir capturar todos os pacotes necessários para análise, podendo, dessa forma, não reconhecer tentativas de ataque;

- ainda não é possível inferir informação alguma sobre o conteúdo de pacotes criptografados. Enquanto as organizações optam pela criptografia para manter a confidencialidade das suas informações, os intrusos a usam para melhorar as chances de sucesso de seus ataques;

- os sensores de rede não conseguem operar bem com os *switches* modernos, pois esses geralmente não possuem portas de monitoramento universal, o que limita bastante a capacidade de captura de pacotes;

- a maioria dos SDI´s baseados em rede não indica se a intrusão foi bem sucedida, só há a indicação que a tentativa de intrusão

começou. Determinar se houve êxito ou não, é importante para o administrador de segurança.

Os sistemas de detecção de intrusão baseados em *host* operam analisando as atividades de algum dos computadores da rede em particular. Isto permite ao SDI coletar informações que refletem o que está acontecendo nos computadores de uma forma bastante precisa. Para tanto, é geralmente utilizado o mecanismo de auditoria do sistema operacional. Assim, o sensor de *host* tem acesso aos diversos *logs* do sistema.

Dentre outras, as vantagens dessa abordagem são que o sistema pode: *i)* monitorar o acesso à informação em termos de "quem acessou o quê"; *ii)* mapear atividades problemáticas com um usuário específico; *iii)* rastrear mudanças de comportamento associadas com mau uso; *iv)* operar em ambientes criptografados; *v)* distribuir a carga do monitoramento ao redor dos diversos computadores da rede.

Dentre as desvantagens encontram-se: *i)* a atividade da rede não é visível pelos sensores de host e, dessa forma, alguns tipos de ataques, não podem ser identificados; *ii)* o acionamento dos mecanismos de auditoria acarreta uma sobrecarga no sistema; *iii)* às vezes, é requerido um espaço em disco considerável para os registros; *iv)* as vulnerabilidades dos sistemas operacionais podem ser usadas para corromper os sensores de host; *v)* os sensores de host são específicos por plataforma, o que acarreta em maior custo no desenvolvimento.

Os SDI´s baseados em aplicação são um subconjunto dos baseados em host, que analisam os eventos ocorridos nas aplicações. A maior fonte de informação são os arquivos de log das transações das aplicações.

A habilidade para interagir com a aplicação diretamente, através do conhecimento do domínio ou aplicação específica incluída na máquina de análise, permite que sejam detectados comportamentos suspeitos, oriundos de usuários tentando exceder o uso de seus privilégios.

Essa abordagem possui algumas vantagens, tais como: monitorar a interação entre usuários e aplicação, permitindo que sejam traçadas atividades desautorizadas para usuários individuais e trabalhar em ambientes criptografados, desde que a interação seja no ponto final da transação, onde as informações são apresentadas aos usuários na forma descriptografada.

As desvantagens são que pode ser mais vulnerável a ataques do que na abordagem baseada em host, pois os logs de aplicação são menos protegidos do que as trilhas de auditoria dos sistemas operacionais. Como os eventos monitorados são a nível de usuário, um SDI, baseado em aplicação, não pode detectar “cavalos de tróia’ ou outro tipo de ataque à aplicação.

Portanto, com base nessa exposição, pode-se concluir que é desejável um SDI que combine as vantagens apresentadas pelas três abordagens, para ter-se dados suficientes para uma análise mais precisa.

2.6 Tempo de coleta e análise dos dados

Dentre as características dos métodos de análise de eventos e os tipos de dados examinados, deve-se considerar particularmente importante o tempo de coleta e análise dos dados.

Na abordagem *batch*, também conhecida como orientada a intervalo, os mecanismos de obtenção de dados guardam os registros em arquivos, que serão periodicamente analisados pelo SDI em busca de sinais de ataques ou mau uso. Desse modo, o fluxo de dados dos pontos de monitoramento não é continuamente analisado.

Esse método é adequado para situações onde a ameaça de ataque é pequena e o interesse maior é saber quem são os intrusos e não tomar medidas imediatas. O modo *batch* requer menos carga de processamento, sendo ideal para instituições que dispõem de recursos limitados. Os ataques geralmente acontecem nos mesmos alvos, em diversas etapas. Nesses casos, o método orientado a intervalo pode detectar as assinaturas de ataque sem maiores problemas.

As desvantagens básicas são que os incidentes na segurança raramente são detectados em tempo hábil para que as devidas providências sejam executadas, maximizando os danos e pode haver a necessidade de grande espaço em disco para armazenar os arquivos gerados, principalmente em ambientes de rede.

Os métodos em tempo real garantem que a análise dos dados coletados será feita continuamente, de forma que o SDI possa agir com a rapidez necessária para frustrar uma tentativa de intrusão qualquer. Dependendo da velocidade da análise, o ataque pode ser detectado rápido o bastante para que ele seja interrompido. Mesmo não sendo possível deter o intruso imediatamente, com uma análise sensível e ágil, os administradores do sistema podem mensurar melhor com os danos causados. É possível, também, coletar as informações necessárias prontamente para que o invasor seja logo penalizado.

No entanto, nesse método, há um consumo de memória e demanda de processamento bem maior que no método *batch*. A configuração de sistemas de tempo real é crítica, pois qualquer assinatura mal caracterizada pode gerar uma enorme quantidade de falsos alarmes em um curto espaço de tempo.

2.7 Resposta aos incidentes

Depois da aquisição dos dados e de sua posterior análise, os sistemas de detecção de intrusos devem reagir às tentativas de invasão ou mau uso. Tal reação pode ser feita de diversas formas, enquadradas em duas categorias básicas: as reações ativas e passivas.

Quando o SDI possui reação ativa, as respostas são dadas automaticamente sempre que determinados tipos de intrusão são detectados. As mais comuns são:

a) Coletar informação adicional

Quando uma tentativa de ataque é detectada, uma boa reação é analisar as evidências com um enfoque mais apurado. Isto pode ser feito, aumentando-se a sensibilidade dos sensores de rede ou *host* para que mais informações possam ser

capturadas e estudadas ou criar armadilhas virtuais (Honeypots). Com mais dados sobre a tentativa de ataque, é possível ter-se mais parâmetros para deter o intruso, ou mesmo, mais provas para investigá-lo e puni-lo sob os limites legais.

b) Alterar as configurações

Em um ambiente de rede, uma solução bastante direta para deter um atacante é simplesmente cortar sua conexão. Isto não é tarefa trivial para um SDI, mas pode ser feito, interagindo-se com o *firewall* e roteadores para tomar algumas providências, entre as quais: bloquear os pacotes de endereço IP de onde o ataque é iniciado; bloquear as portas, protocolos ou os serviços usados pelos atacantes ou, em caso extremo, fechar todas as conexões que usam determinadas interfaces de rede.

c) Executar ações diretas contra o intruso

A ação mais agressiva que se pode tomar em caso de intrusão é atacar o atacante. Assim, o *hacker* sofrerá com as estratégias que ele próprio usou. Uma abordagem menos hostil é tentar obter informações do computador ou *site* do intruso, para posterior tomada de medidas legais.

O problema de reagir dessa forma tão enérgica é que provavelmente alguma lei é infringida. Assim como é ilegal alguém invadir os sistemas de uma instituição, também o é responder de forma semelhante. Outro problema é que os ataques geralmente partem de endereços de rede falsos, fazendo com que aumente o risco de que *sites* ou usuários inocentes sejam prejudicados.

As respostas passivas delegam ao pessoal de segurança a tarefa de tomar as providências necessárias em relação ao atacante. A reação mais comum é disparar alarmes.

Quando um ataque é detectado, notificações podem ser geradas pelo SDI para avisar aos usuários. A forma mais comum de alarme é um aviso na tela, entretanto, em grandes organizações, é comum a notificação de alertas remotos. A informação provida nas mensagens pode variar bastante, indo de uma simples notificação da intrusão até mensagens detalhadas sobre o grau de severidade do

ataque, especificando seu tipo, o endereço IP de onde partiu, o alvo e as técnicas utilizadas.

2.8 Sistemas de detecção de intrusos baseados em agentes

Atualmente, os sistemas de detecção de intrusos que utilizam arquiteturas monolíticas, enfrentam problemas referentes à:

i) tolerância a falhas - um SDI monolítico possui um ponto de falha, que pode ser facilmente atingido através de ataques, do tipo negação de serviço (DoS), ao servidor o qual o sistema está instalado;

ii) dificuldades em reconfigurar ou adicionar capacidades ao sistema em tempo real;

iii) eficiência - Os SDI´s precisam ser reconstruidos para poderem detectar novas formas de intrusão que não estejam previstas no sistema;

iv) necessidade do uso de pessoal técnico qualificado e de ferramentas para manter o conhecimento;

v) escalabilidade;

De acordo (Crosbie e Spafford, 1995; Zamboni et al, 1998) segue-se que, a partir dessas desvantagens, existem alguns fatores motivadores quanto ao uso de agentes inteligentes (Apêndice A) para detectar intrusões, os quais venham garantir as características desejáveis em um SDI:

i) os agentes podem ser adicionados e removidos do sistema sem precisar reiniciá-lo, tornando fácil a sua manutenção e atualização;

ii) os agentes podem ser treinados para identificar novos ataques;

iii) os agentes podem ser ativados e desativados dinamicamente para realizarem suas tarefas, proporcionando, com isso, melhor uso dos recursos do sistema;

iv) um agente pode ser configurado especificamente para as necessidade de um host ou uma rede, aumentando o poder de configuração do sistema;

v) o uso de agentes garante maior tolerância a falhas do que em sistemas monolíticos. Segundo (Negroponte, 1997), "*uma estrutura descentralizada e dotada de alto grau de intercomunicação apresenta flexibilidade bem maior, e maior probabilidade de sobrevivência. Ela será de certo mais sustentável e é mais provável que evolua ao longo do tempo*";

vi) agentes podem ser adicionados para aumentar a capacidade do SDI, permitindo que os mesmos operem em sistemas maiores.

Visando atingir essas características, o uso de agentes em sistemas de detecção de intrusão tem sido proposto por diversos laboratórios de pesquisa no desenvolvimento de sistemas não-monolíticos, dentre eles, a Universidade Purdue (Zamboni et. al, 1998), a Escola de Pos-graduação Naval, Monterey, CA (Barrus e Rowe, 1998), a Universidade de Idaho (Frincke et al., 1998), a Universidade Columbia (Lee et al., 1999), a Universidade Brandenburg de Tecnologia (Sobirey e Richer, 1999), a Universidade do Estado de Iowa (Helmer et al, 1998), a Agência de Promoção de tecnologia da informação, Japão (Asaka et al., 1999) e o Instituto de Ciências Matemáticas de São Carlos - ICMC/USP (Cansian et al, 1997a, 1997b, 1997c; Bonifácio Jr. et al, 1998). A Tabela 2.1 apresenta uma análise comparativa relacionada às características relevantes das arquiteturas mencionadas.

Tabela 2.1 – Comparativo entre protótipos de SDI´s

Característica / Protótipo	Método de Análise		Tipo de Análise		Tempo de coleta		Resposta		Plataforma Operacional	
	Abuso	Anomalia	Rede	Host	Real	Batch	Ativa	Passiva	Unix[23]	Win
AAFID2	X		X	X	X			X	X	
IDAgent	X		X	X	X			X		X
Hummingbird	X			X	X			X	X	
JAM	X			X	X			X	X	
SADI	X		X		X		X	X	X	
IDA	MLSI			X	X		X	X	X	
AID	X			X	X			X	X	

A arquitetura conhecida como “Agentes Autônomos para Detecção de Intrusão” (AAFID), desenvolvido pelo CERIAS - *Center of Education and Research in Information Assurance and Security at Purdue University* (Zamboni et. al, 1998), é um SDI clássico que utiliza uma estrutura hierárquica, composta por agentes, *tranceivers* e *monitors*.

Nessa arquitetura, os agentes, responsáveis por capturar as informações dos servidores, residem em uma plataforma de agente chamada *transceivers*. Eles controlam, analisam e processam as informações recebidas pelos agentes. Nesse sistema, existe um *transceiver* para cada servidor monitorado. Os *monitors* são responsáveis pelo controle global do sistema. Eles controlam, analisam e processam as informações recebidas dos servidores monitorados, detectando, dessa forma, eventos suspeitos ocorridos em diferentes servidores. Esse protótipo, que atualmente está na sua segunda versão (AAFID2), foi implementado em Perl, para ser executado em ambientes Unix.

A Escola de Pós-graduação Naval (Barrus e Rowe, 1998) apresentou uma proposta de arquitetura distribuída com agentes autônomos para monitorar, em tempo real, atividades relacionadas à segurança em rede. Esse projeto prevê ainda

[23] O termo Unix é utilizado de forma genérica, visto que existem diversos sistemas operacionais baseados nessa plataforma, por exempo o Solaris, AIX, HP-UX, Linux, etc.

o uso de redes neurais para medir e determinar o nível de alerta que é repassado, através de mensagens, para os demais agentes em caso de um ataque. O protótipo desenvolvido (*IDAgent*) para operar em ambiente Windows NT 4.0, foi implementado usando-se as linguagens Java e C na implementação do *Log Sensor*.

A Universidade de Idaho desenvolveu o projeto *Hummingbird* (FRINCKE et al., 1998). Esse é um protótipo de detecção de intrusão distribuído que utiliza o método de detecção por abuso, utilizando a arquitetura de agentes. Porém, apesar de não utilizar agentes autônomos, as suas ferramentas, algoritmos, redução de dados e técnicas de visualização oferecem uma promessa considerável para o uso da arquitetura de agentes móveis.

A Universidade do Estado de Iowa (Helmer et al, 1998) desenvolveu um SDI baseado em agentes inteligentes, usando a colaboração e a tecnologia de agentes móveis e aplicando tecnicas de *data mining* para identificar e reagir a ataques coordenados em multiplos subsistemas. Esse projeto possui algumas semelhanças ao projeto JAM (Java Agents for Meta-Learning) da Universidade Columbia, NY (LEE et al., 1999).

O projeto JAM aplica meta-aprendizagem aos dados distribuídos e utilizam agentes inteligentes para detecção de intrusão. Os agentes inteligentes empregam técnicas de inteligência artificial para modelar o conhecimento, bem como o comportamento, em sociedades multiagentes ou domínios. O sistema possui dois componentes: os agentes detectores de fraude locais que aprendem como detectar fraudes e promover o serviço de detecção de intrusão dentro de um único sistema de informação corporativo e um sistema de meta-aprendizagem integrado que combina o conhecimento coletivo adquirido pelos agentes locais.

O projeto da Universidade Brandenburg de Tecnologia em Cottbus, Alemanha (The Intrusion Detection System - AID) é uma arquitetura cliente-servidor para sistemas Unix, constituída por agentes instalados em servidores e uma estação de monitoramento central (SOBIREY; RICHER, 1999).

A Agência de Promoção de Tecnologia da Informação (IPA), no Japão, está desenvolvendo um SDI chamado Sistema Agente de Detecção de Intrusão -

IDA (Asaka et al., 1999). O IDA utiliza agentes móveis para localizar eventos específicos em servidores que possam ser relacionados a intrusões, conhecidos como MLSI (Marks Left by Suspected Intruder).

O Instituto de Ciências Matemáticas de São Carlos (ICMC/USP) desenvolveu um SDI baseado em redes que usa técnicas de filtragem e captura de pacotes e redes neurais para detectar eventos suspeitos. O sistema SADI (Sistema Adaptativo de Detecção de Intrusão) é composto por agentes de segurança instalados em servidores seguros (Unix) e distribuídos em pontos estratégicos da rede (Cansian et al, 1997a, 1997b, 1997c) (Bonifácio Jr. et al, 1998). Em 1999, Mauro Bernardes (Bernardes, 1999) propôs uma arquitetura em camadas, composta por agentes móveis na detecção de intrusos.

2.9 Considerações finais

Neste capítulo foram apresentados os conceitos e as características dos sistemas de detecção de intrusos. Discutem-se as razões para o emprego da tecnologia de agentes aplicada na detecção de intrusos, mostrando-se as suas vantagens sobre os sistemas monolíticos.

De fato, a abordagem de agentes é desejável, pois fornece robustez, confiabilidade e flexibilidade a um SDI, promovendo execução contínua, tolerância a falhas, resistência à subversão, sobrecarga mínima e elevado nível de escalabilidade e configuração, permitindo a inserção e remoção de agentes de acordo com o cenário de atuação e descoberta de novas tecnologias e conhecimentos de prevenção e proteção de sistemas computacionais.

Foram mostradas algumas pesquisas recentes sobre SDI, utilizando essa tecnologia, constatando-se que a grande maioria foi desenvolvida para operar em plataforma Unix. Porém, segundo pesquisas recentes (Attrition, 2001), o sistema operacional que indica estar sujeito ao maior número de incidentes é o Windows

NT/2000[24] e isso faz com que SDI para esse sistema seja uma boa área a ser explorada.

Portanto, a continuidade dos projetos e a busca de métodos alternativos de segurança de redes têm sido uma preocupação constante dos pesquisadores. Nesse contexto, a presente proposta representa mais uma possibilidade de expansão do conhecimento teórico e prático de novas tecnologias aplicadas a essa área. O correspondente detalhamento será apresentado no Capítulo 3.

[24] O Windows NT possui aproximadamente 60% de registros de incidentes de segurança e ao contrário do que pode parecer, essa plataforma não é a mais utilizada na Internet (ALLDAS, 2002).

3 PROPOSTA DE UM MODELO DE AGENTES PARA DETECÇÃO DE INTRUSOS

Neste capítulo é proposto um modelo de sistema de detecção de intrusos em rede de computadores, utilizando-se a tecnologia de agentes inteligentes, com o objetivo de explorar as características desejáveis apresentadas por vários SDI´s existentes na literatura. Apresentar-se-á a arquitetura e funcionalidade dos agentes em sociedade, fornecendo uma visão de modo a delimitar e justificar a contribuição deste trabalho.

3.1 Considerações iniciais

A proposta do modelo de sistema de detecção de intrusos apóia-se em diversas arquiteturas disponíveis, a fim de buscar melhores soluções para os problemas enfrentados na implementação de um sistema desse nível. Com isso, buscou-se modelar um sistema que abordasse as principais características desejáveis, pricipalmente quanto às respostas aos incidentes, visto que muitos sistemas não possuem respostas ativas às tentativas de intrusão.

O modelo proposto prevê a metodologia de detecção por abuso e anomalia para garantir uma robustez maior ao sistema. Entretanto, para efeito de ilustração, utilizou-se a detecção por abuso como método de análise. A escolha se deu em virtude da grande maioria dos ataques poderem ser codificados, de maneira a capturar e registrar variantes a cerca de atividades que exploram as mesmas vulnerabilidades. Para isso, diversos agentes são responsáveis pelas funções de monitoramento, análise dos dados, detecção e resposta às atividades suspeitas.

No monitoramento, o sistema adotou uma combinação de agentes sensores que serão instalados em pontos estratégicos da rede e em *hosts* críticos com o objetivo de capturarem pacotes suspeitos e atividades maliciosas para serem analisadas.

Os agentes coletam e analisam as informações em *batch* ou em tempo real, dependendo do ambiente e da política de segurança da organização. Os fatores que mais influenciam essa escolha são o nível de risco associado, o consumo de memória e recursos disponíveis e o papel que o próprio sistema de detecção deve desempenhar.

Os tipos de reações a serem tomadas podem variar, de acordo com a segurança da corporação, tendo-se obrigatoriamente, respostas *default* no caso de detecção de novos ataques. As reações são as mais diversas, podendo-se simplesmente enviar notificações ao administrador ou reagir de forma mais ativa, coletando-se informações adicionais do atacante (através de um ambiente virtual – *Decoy Server*), alterando-se as configurações do ambiente ou realizando-se ações diretas contra o intruso.

Um dos propósitos do modelo é possuir a capacidade de interagir com sistemas tipo *firewall*, no sentido de diminuir os problemas apresentados, permitindo que seja alcançado um nível de segurança desejado, uma vez que os dois sistemas possuem características complementares. O detalhamento dessa interação será assunto de trabalhos futuros.

Outras características importantes do sistema proposto são que: *i)* o modelo é capaz de identificar intrusões que nunca tenham ocorrido, através de sua capacidade de generalização, além de poder-se adicionar novos padrões de ataques à medida que forem surgindo, com a utilização de uma rede neural; *ii)* ele pode adaptar-se às estratégias de segurança da organização, fornecendo informações a respeito de tentativas de intrusão, bem como as ações tomadas, permitindo que o administrador de segurança tome conhecimento do que está ocorrendo na rede e *iii)* ele é projetado para operar nos ambientes Unix e Windows NT.

3.2 Arquitetura geral do sistema

A arquitetura do sistema proposto é composta pela cooperação entre agentes artificiais (*Agentes Sensores*, *Agente de Monitoramento, Agente de Avaliação de Segurança*, *Agente de Atualização, Agente Controlador de Ações*, *Agente de Integridade e Agente de Segurança Local*) e agentes humanos (*Administrador de Segurança*) (Figura 3.1).

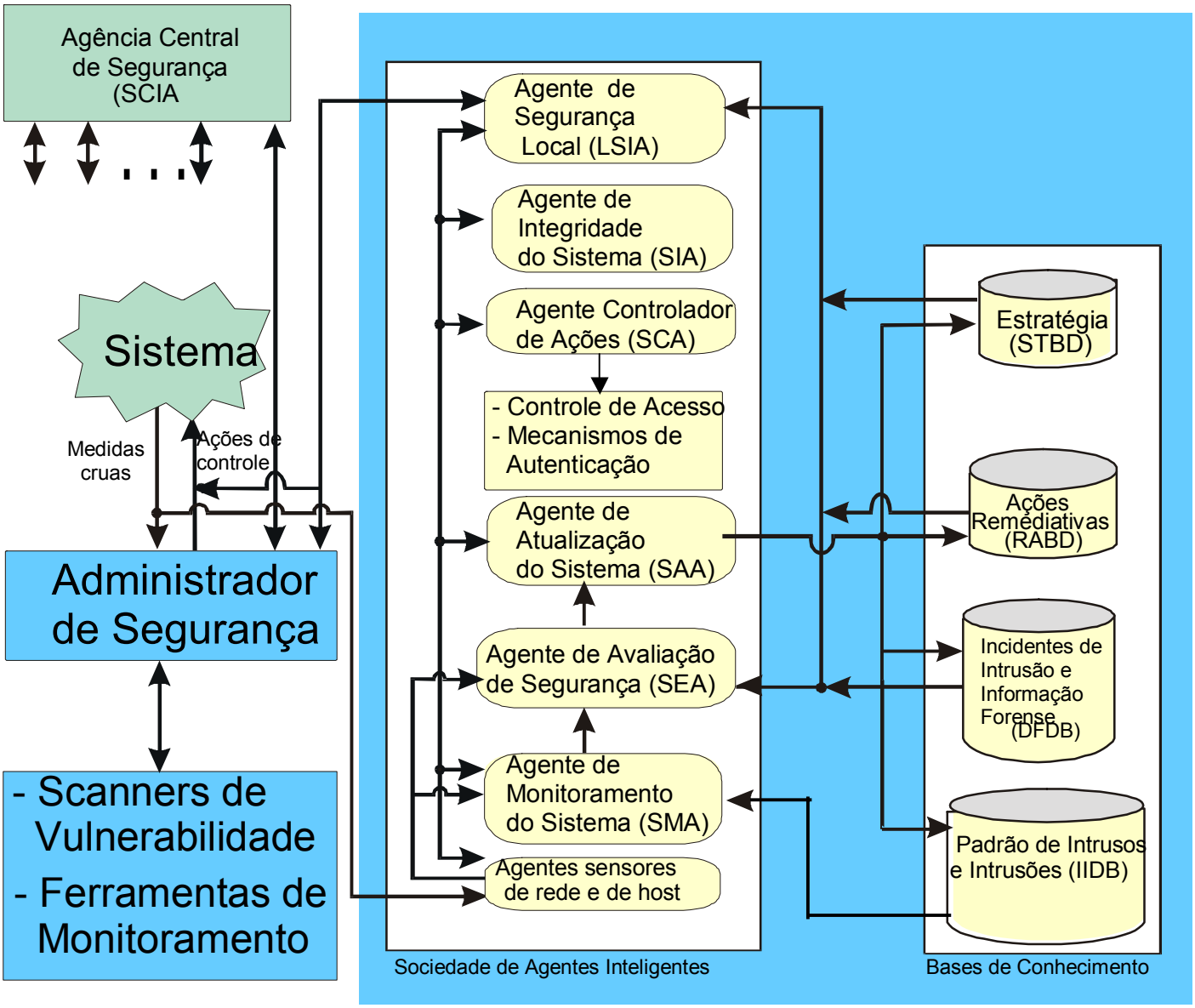


Figura 3.1 - Arquitetura Geral do SDI proposto

A seguir, descrevem-se as responsabilidades de cada agente (artificial e humano) na arquitetura proposta:

Agentes Sensores: esses agentes são responsáveis por capturar as informações que estão trafegando na rede e no servidor e enviá-las para o agente de monitoramento (SMA). O modelo propõe o uso de dois tipos de agentes sensores: **os agentes sensores de rede** e os **agentes sensores de *host***. Os agentes sensores de rede são responsáveis pela captura dos pacotes que estão trafegando na rede monitorada (*sniffing*), enquanto que os agentes sensores de *host* coletam informações em arquivos de *log* em *hosts* críticos.

Agente de Monitoramento de Sistema (System Monitoring Agent - SMA): é responsável por organizar e formatar os eventos ou conjunto de eventos coletados, de forma que possam ser identificados padrões de ataques e comportamentos anormais na rede e servidores monitorados, de acordo com a base de dados de padrões de intrusões (IIDB) e atribuir um grau de risco representado pelo evento detectado.

Agente de Avaliação de Segurança do Sistema (System Security Evaluation Agent - SEA): é o agente responsável por verificar se o evento corresponde, de fato, a uma tentativa de invasão ou trata-se apenas de um falso positivo. Para auxiliar nessa tarefa, ele faz uso de informações das bases de dados de incidentes de intrusão (DFDB) e estratégias (STBD). Como saída, esse agente gera um nível de alerta associado à severidade do evento.

Agente de Atualização do Sistema (System Atualization Agent - SAA): é responsável por manter atualizadas as bases de dados de incidentes de intrusão (DFDB), de padrões de intrusões (IIDB), de ações (RABD) e estratégias (STBD).

Agente Controlador de Ações (System Controller Agent - SCA): esse agente é responsável pelo controle das ações que o sistema deve tomar em caso de uma tentativa de invasão. Para a tomada de decisão, são utilizadas as bases de dados de estratégia (STBD) e ações (RABD).

Agente de Integridade do Sistema (Self-Integrity Agent - SIA): esse agente é responsável por garantir a integridade do SDI. O agente de integridade busca por eventos não esperados ou diferentes do perfil normal dos agentes ativos do sistema.

Agente de Segurança Local (Local Security Intelligent Agent - LSIA): esse agente é responsável pelo gerenciamento da sociedade de agentes e pela interface entre o SDI com o administrador de segurança. É através desse agente que o administrador gerencia o status e a configuração dos agentes, a atualização das bases de dados, o registro das ocorrências detectadas e as ações tomadas pelo sistema.

Agente Administrador de Segurança: é o agente humano que interage com o SDI através do agente de segurança local para realizar diversas tarefas, tais como: configurar a política de segurança do ambiente computacional (base de dados de estratégias) e as demais bases de dados; ativar e desativar agentes; gerenciar o *status* e a configuração do sistema, além de conhecer a situação atual do ambiente monitorado.

Bases de Conhecimento: o sistema dispõe de quatro repositórios para armazenar as informações relevantes à detecção de intrusão. O STBD é a base de dados responsável por registrar as estratégias adotadas por uma organização qualquer em relação à sua política de segurança. Ela é importante para garantir a adaptabilidade do SDI aos mais diversos casos. No RABD estão contidas as informações referentes às ações que devem ser tomadas de acordo com a severidade do ataque detectado. Também varia de acordo com a política de cada instituição. O IIDB guarda as assinaturas de intrusão que serão utilizadas para a detecção de atividades suspeitas. Ele deve ser constantemente atualizado para garantir que novas técnicas de ataque possam ser detectadas. Por fim, tem-se o DFDB, que registra os danos causados por ataques bem-sucedidos e tentativas de ataques. Nele ficam contidas informações que podem ser úteis na identificação de tentativas de ataques provenientes de uma mesma origem ou domínio ou simplesmente para serem utilizadas em investigações futuras.

A estrutura do modelo apresentada foi exposta somente no primeiro nível do domínio, não sendo explorados os diversos níveis de profundidades existentes, conforme mostra a Figura 3.2.

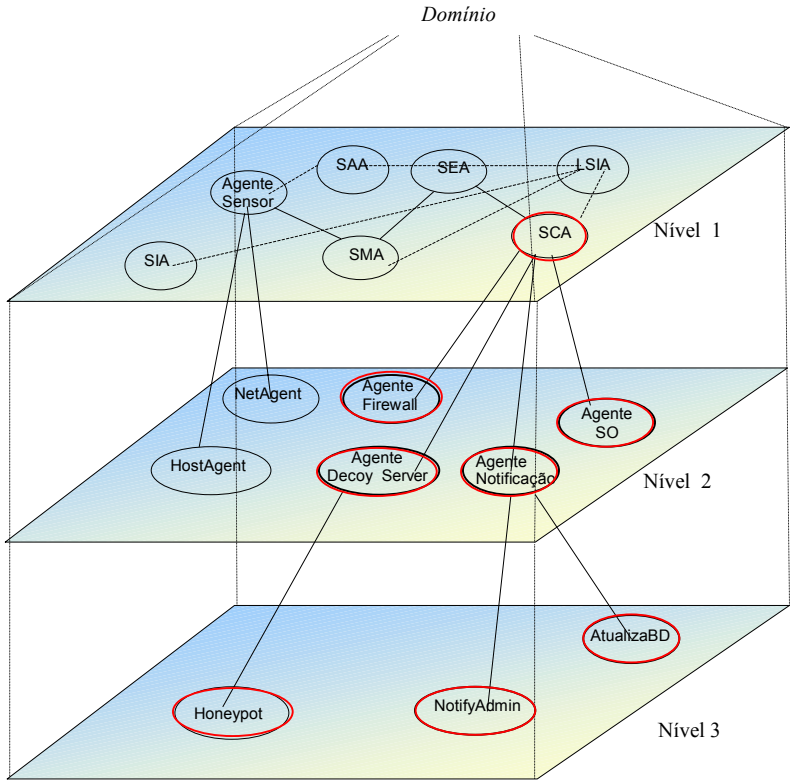


Figura 3.2 – Aquitetura NIDIA em profundidade.

A exemplo disso, tem-se o Agente Controlador de Ações (SCA) que, ao ser acionado, poderá ativar outros agentes para aplicarem as medidas necessárias contra as tentativas de ataques (SANTOS, 2002). Nesse contexto, o Agente Decoy Server, responsável por criar um ambiente virtual, poderá ser ativado, permitindo que o sistema possa obter maiores informações a respeito do intruso, ativando o Agente Honeypot (OLIVEIRA, 2002). O Agente Firewall e o Agente SO poderão ser acionados para reconfigurar o ambiente (alterar filtros de pacotes, mudar permissões de usuários, grupos e objetos) para impedir que novas tentativas de ataques sejam efetuadas. O Agente de Notificação também aciona diversos agentes, pertencentes a níveis de profundidade diferentes (Figura 3.2), para realizar a sua tarefa.

3.3 Funcionamento genérico do SDI

Para um melhor entendimento da arquitetura proposta, é explicado o funcionamento genérico do sistema de detecção de intrusão. Isto é feito com o auxílio de um diagrama de colaboração, em notação UML, apresentado na Figura 3.3.

Inicialmente, os agentes sensores de *host* e de rede capturam os dados do sistema de *log* de um computador isolado e do tráfego da rede, respectivamente. As informações relevantes são passadas ao Agente de Monitoramento do Sistema (SMA). Esse agente efetua uma pré-análise dessas informações, verificando se os eventos ocorridos podem conter indícios de ataque, comparando-os a uma base de padrões de intrusão (IIDB). O processamento pode ser feito através de uma rede neural, o que garante uma grande flexibilidade ao processo, pois mesmo que os eventos ocorridos não sejam exatamente iguais aos do IIDB, a tentativa de intrusão pode ser detectada. Uma vez atingido um limite máximo de suspeita, tais informações são repassadas ao Agente de Avaliação de Segurança do Sistema SEA).

Então, o SEA faz uma análise de alto nível das informações, levando-se em consideração variáveis, contidas nas bases de dados de intrusões e estratégias, verificando se o evento corresponde realmente a um ataque. Essas variáveis podem ser: o horário em que a ação está sendo feita; o domínio ou computador de origem do evento; o usuário que está efetuando a ação etc. Se o SEA considerar que a segurança do sistema realmente está em perigo, então o Agente Controlador de Ações (SCA) é avisado.

O SCA, de acordo com as informações recebidas, analisa o grau de severidade do ataque e propõe as medidas cabíveis. Depois, o SCA solicita que atualizações sejam feitas, a fim de registrar as informações sobre o ataque no DFDB. Isto é importante para que se tenha informação das origens das tentativas de intrusão e intrusões consumadas, bem como os danos causados e o histórico dos incidentes de segurança. De acordo com a estratégia adotada por cada empresa, uma ação automática é disparada ou o administrador do sistema é notificado.

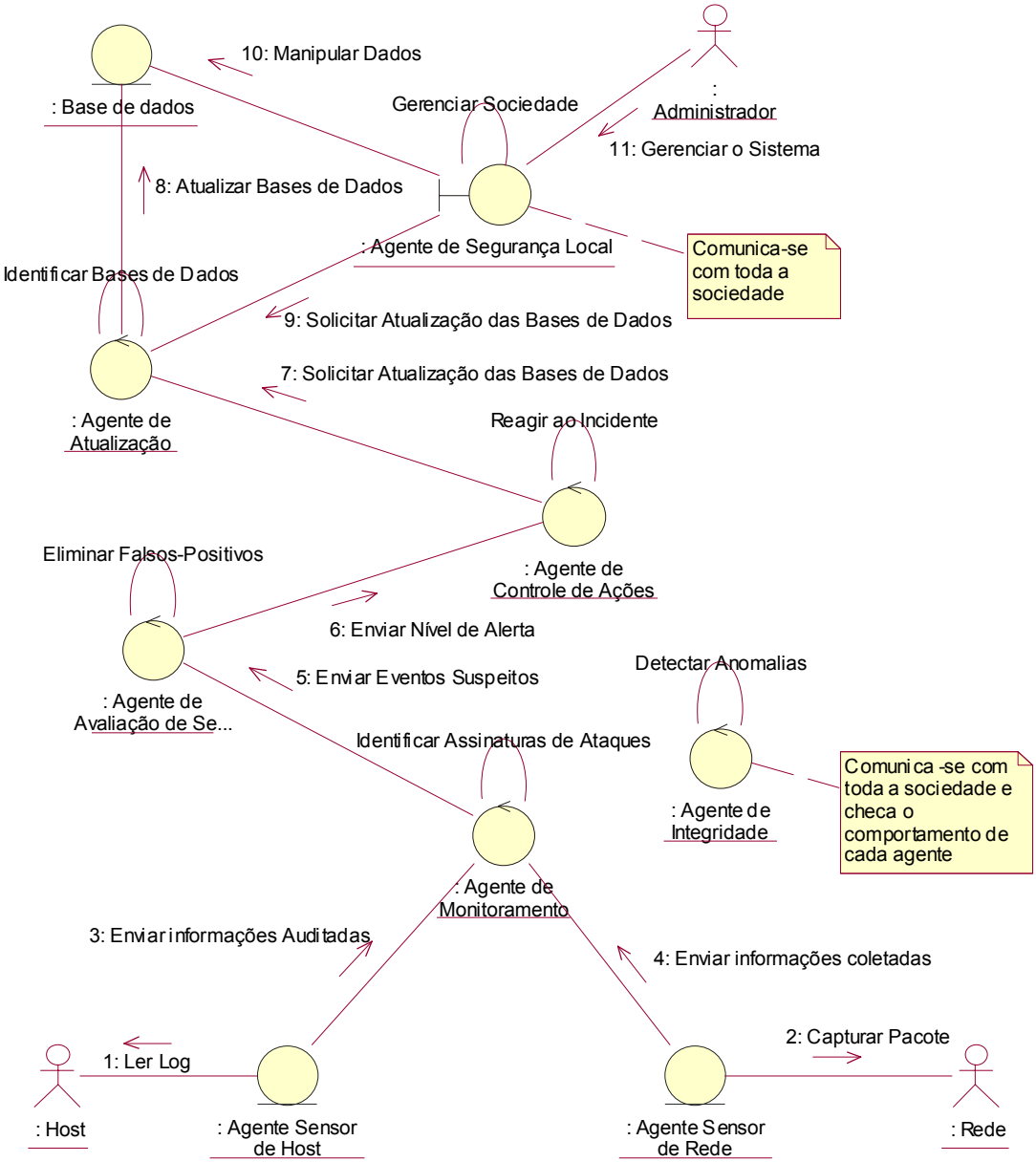


Figura 3.3 – Diagrama de Colaboração do SDI proposto (notação UML).

Para que as atualizações nos bancos de bancos sejam feitas, é necessária a intervenção do Agente de Atualização do Sistema (SAA). Ele atualiza os quatro bancos de dados quando da solicitação de algum agente ou do administrador.

Por fim, tem-se o Agente de Segurança Local (LSIA), que é responsável pelo gerenciamento dos demais agentes, verificando o estado do SDI como um todo. Se algo estiver errado, a equipe da segurança deve ser informada. Aliás, esse é o único agente que se comunica diretamente com o administrador do sistema, realizando as mais diversas tarefas de controle.

Outro agente que se comunica com todos os outros adentes da sociedade é o Agente de Integridade do Sistema (SIA). Sua função é verificar se o próprio SDI foi alvo de ataque. Para isso, o perfil de cada agente é levantado com os respectivos limites de normalidade. Se essa barreira for ultrapassada por algum agente, é bastante provável que o sistema esteja corrompido e, portanto, medidas devem ser tomadas, assim como, o administrador deve ser informado.

Convém ressaltar que, como em qualquer sistema multiagentes, os eventos do modelo não ocorrem de forma seqüencial, conforme foi mostrado, pois cada agente é executado em uma linha de execução independente e de forma distribuída (em vários servidores e estações de trabalho), competindo e cooperando entre si para realizarem tarefas, de forma a proteger o ambiente computacional sem, preferencialmente, gerar impacto negativo no desempenho da rede.

Foram omitidas do contexto, as estratégias de negociação utilizadas na interação dos agentes, discutidas no Apêndice A e os níveis de profundidade presentes na arquitetura para que fossem concentrados esforços em dar uma visão geral de funcionalidade do sistema em sociedade, através de um cenário demostrativo de colaboração.

3.4 Principais casos de uso do modelo

A seguir, serão apresentados os principais Casos de Uso e Atores, para descrever e definir os requisitos funcionais do SDI proposto. Inicialmente serão abordadas as responsabilidades do administrador de segurança dentro do modelo de detecção de intrusão proposto, conforme mostrado na Figura 3.4.

O papel do administrador de segurança é de vital importância para uma rede de computadores, visto que o mesmo é responsável por diversas tarefas que

exigem um conhecimento técnico elevado e dedicação exclusiva à administração de segurança de uma corporação.

É ele que interage com o Sistema de Detecção de Intrusão para alimentá-lo com informações essenciais (por ex. novos padrões de ataques, estratégias para a tomada de decisão em caso de incidentes) à eficiência do SDI; ativar e desativar agentes, dependendo das necessidades de segurança da organização (por ex. em caso de expansão ou modificação da estrutura da rede) ou por motivos, tais como análise de desempenho; gerenciamento do *status* e da configuração do sistema.

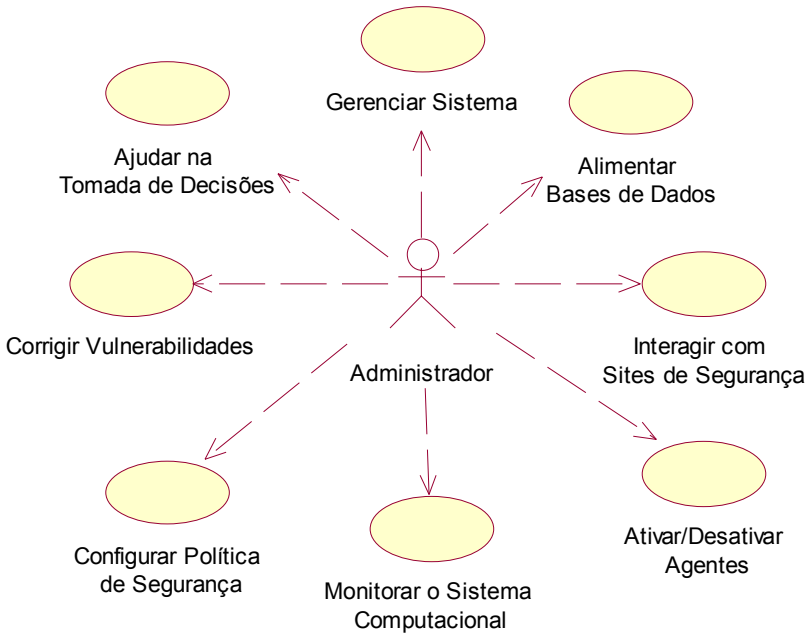


Figura 3.4 – Principais Casos de Uso do Administrador de Segurança (notação UML)

É necessário que o administrador mantenha-se atualizado quanto à identificação e correção de vulnerabilidades, através de *sites* especializados de segurança (por ex. www.cert.org, www.securityfocus.com, www.modulo.com.br), além de ser responsável por configurar o SDI para trabalhar conforme a política de segurança da organização e poder monitorar, permanentemente, o acesso aos servidores e serviços do sistema computacional, pois um intruso pode permanecer dias dentro de de uma rede sem que seja detectado.

O administrador possui muito mais funções do que as citadas, porém será dada ênfase as tarefas relacionadas com o contexto do trabalho. É importante enfatizar que segurança não é somente uma questão técnica, mas uma questão gerencial e humana, pois essa problemática não se resolve somente adquirindo hardware e software, sem que haja um envolvimento de todos os níveis hierárquicos da empresa.

A seguir, será mostrado o diagrama de seqüência dos agentes sensores, iniciando-se pelo agente sensor de *host*, apresentando-se a sua interação com o SDI (Figura 3.5).

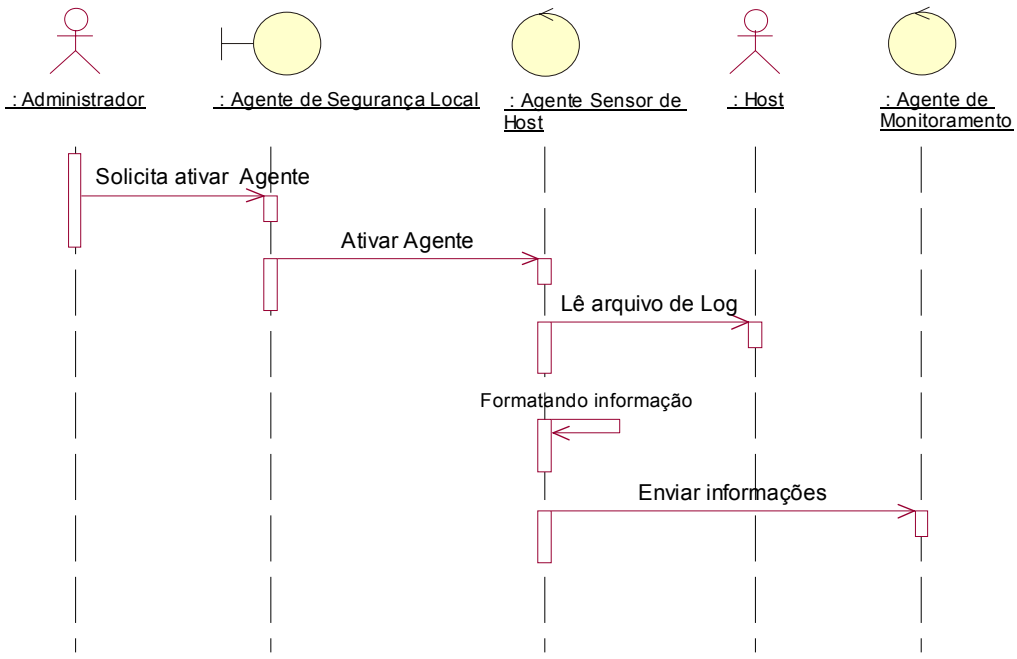


Figura 3.5 –Diagrama de Seqüência do Agente Sensor de *Host* (notação UML)

Conforme descrito anteriormente, é através dos agentes sensores de *host*, que o SDI examina os *logs* de auditoria ou registros de tráfego de rede dos *hosts* críticos para identificar intrusões. Esse agente é responsável por coletar atividades suspeitas em logon, manipulação (criação, alteração e deleção) de grupos, usuários e objetos, privilégios de processos e usuários, *restart/shutdown* e uso de recursos do sistema monitorado.

O diagrama de seqüência do agente sensor de rede é mostrado na Figura 3.6. O cenário desse agente é semelhante ao agente sensor de *host*, diferindo somente no tipo de informações coletadas (Apêndice B).

Através dos agentes sensores de rede instalados em segmentos estratégicos da rede monitorada, o SDI examina o conteúdo dos pacotes, procurando por assinaturas de ataques.

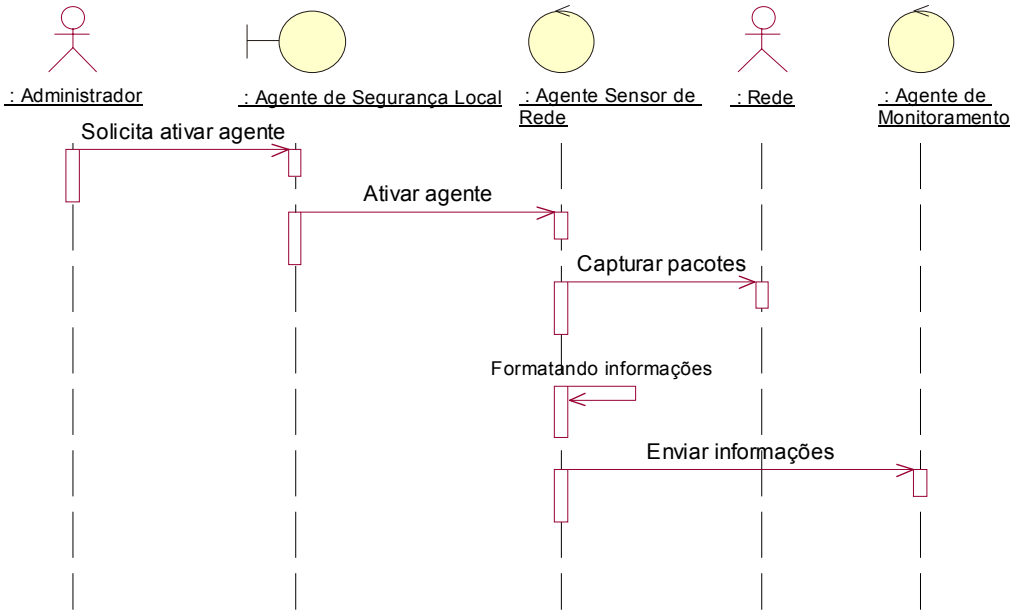


Figura 3.6 – Diagrama de Seqüência do Agente Sensor de Rede (notação UML).

Quando o sistema é inicializado, o administrador informa ao SDI, através do agente inteligente de segurança local, quantos e quais agentes sensores (de rede ou de *host*) serão ativados, tomando como base a arquitetura de rede da organização (ativos[25]).

Ao serem ativados, os agentes sensores começam a coletar os dados para serem avaliados se representam uma atividade maliciosa. Em operação

[25] Ativos são elementos aos quais a organização atribui valor e desta forma requerem proteção. O valor atribuído a cada ativo está relacionado à confidencialidade, integridade e disponibilidade do mesmo.

normal, vários agentes sensores são ativados e desativados da sociedade, automaticamente, para atingir as metas estabelecidas pelo SDI.

Esses agentes têm uma característica importante, que é de poderem ser configurados para coletar dados específicos, permitindo a construção de vários agentes sensores de *host* e de rede com habilidades diferentes.

Os agentes sensores geram informações, num formato padrão que, a priori, não expressam claramente uma evidência de ataque e, portanto, necessitam de uma análise mais aprofundada. O agente que possui a habilidade de receber e entender tais informações é o agente de monitoramento (Figura 3.7), que usa a base de dados de padrões de intrusão (IIDB) para identificar assinaturas de ataques que podem corresponder a uma tentativade intrusão ou até mesmo uma invasão consumada.

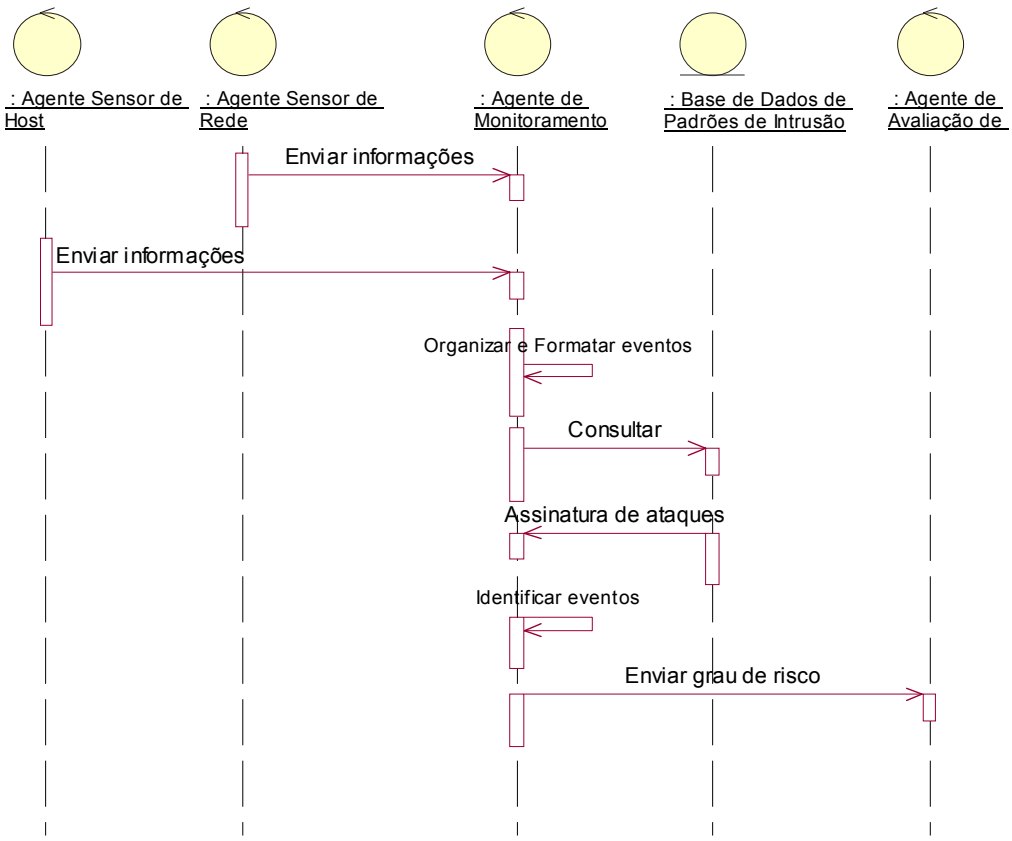


Figura 3.7- Diagrama de Seqüência do Agente de Monitoramento (notação UML)

O agente de monitoramento identifica padrões de ataque, através de uma rede neural, comparando-as com a base de dados de padrões de intrusão (IIDB). Esse agente é dotado da capacidade de generalização, ou seja, novos ataques poderão ser detectados, desde que apresentem semelhanças aos padões pré-existentes.

Em seguida, os eventos considerados suspeitos são enviados, com um grau de risco associado, para que possam ser analisados por um agente mais especializado, que no caso, é o agente de avaliação de segurança (Figura 3.8).

O agente de avaliação de segurança é responsável por verificar a autenticidade dos resultados obtidos pelo agente de monitoramento baseados em um conjunto de dados disponíveis nas bases de dados de incidentes de intrusão (DFDB) e estratégias (STBD). Como saída, esse agente gera um nível de alerta associado ao grau de risco do evento detectado, que é posteriormente interpretado pelo agente controlador de ações.

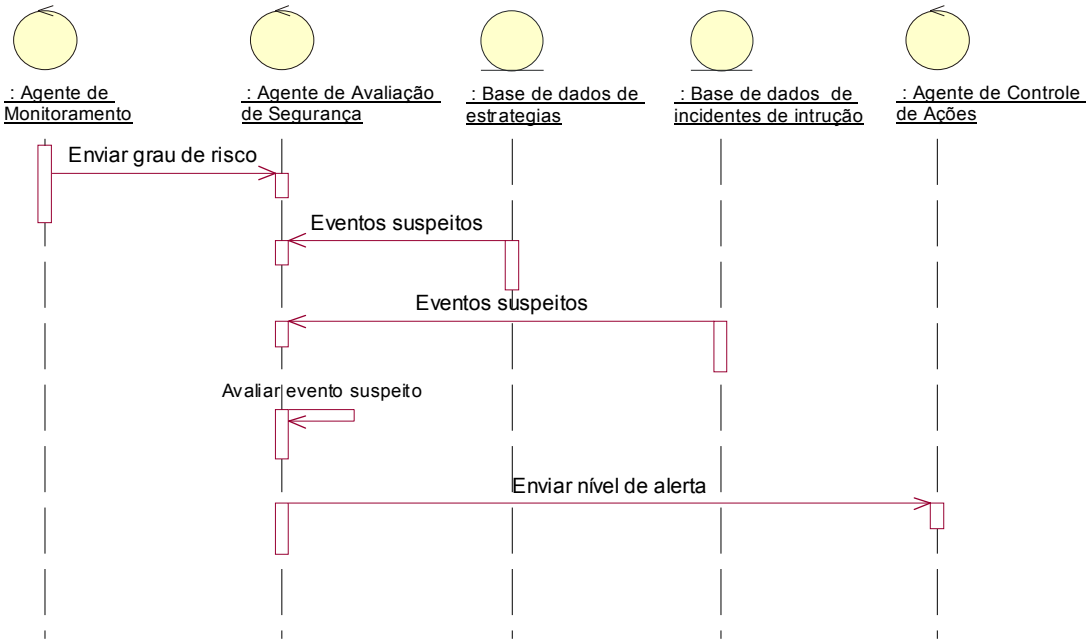


Figura 3.8 – Diagrama de Seqüência do Agente de Avaliação de Segurança.

O Agente Controlador de Ações é responsável pelas respostas aos incidentes detectados. A ação, fornecida pela base de dados de ações, será utilizada conforme o nível de alerta gerado pelo agente de avaliação e o tipo de estratégia adotada (STBD), ou seja, esse agente poderá solicitar a ativação de outros agentes sensores para coletar mais informações das conexões suspeitas (para garantir que nenhum evento importante seja descartado acidentalmente) ou tomar atitudes mais radicais, como quebrar uma conexão, remover ou alterar as permissões de usuários, reconfigurar tabelas de roteadores e firewall, ou simplesmente solicitar ajuda ao administrador em caso de dúvida na tomada de decisão.

Esse agente deverá registrar tais ocorrências (evento detectado + ação tomada) para futuras análises e correções de vulnerabilidades ou com o propósito de um suposto processo judicial, bem como interagir com o agente de segurança local para atualizar as bases de dados envolvidas nesse cenário, quando forem tomadas novas ações ou forem detectados novos ataques (Figura 3.9).

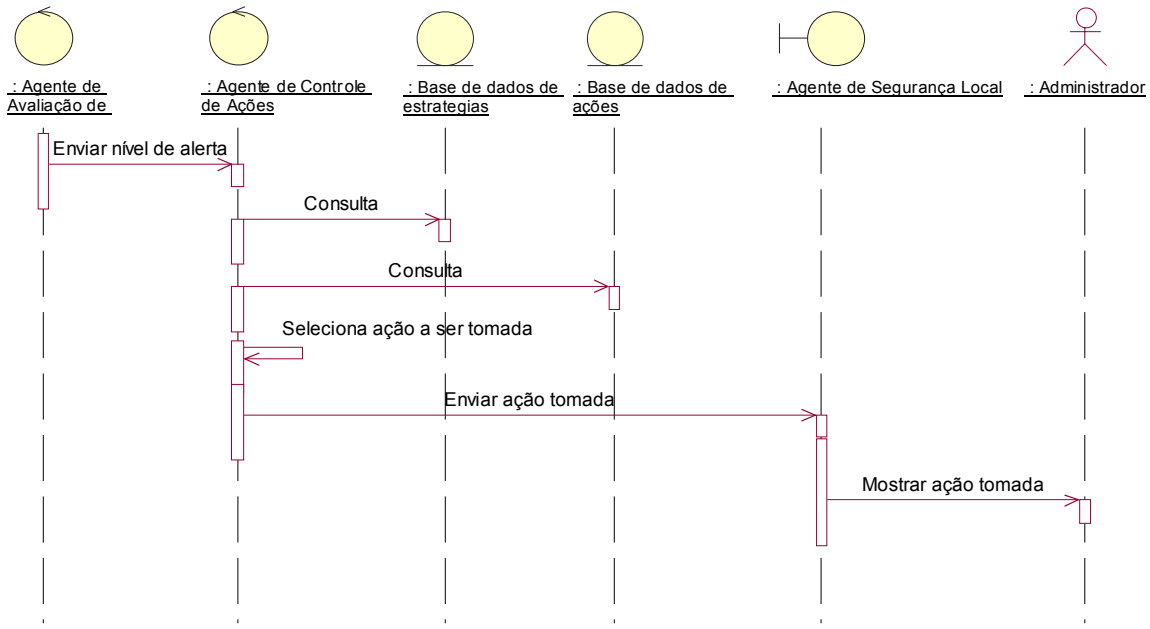


Figura 3.9 – Diagrama de Seqüência do Agente Controlador de Ações (notação UML)

O Agente de Atualização mantém atualizadas as bases de dados de incidentes de intrusão (DFDB), de padrões de intrusões (IIDB), de ações (RABD) e estratégias (STBD). Esse agente pode ser ativado no caso de ser detectada uma nova intrusão pelo sistema ou o administrador de segurança necessitar atualizar o sistema, após a divulgação de novos ataques ou descoberta de novas vulnerabilidades.

O Agente de Atualização interage com o agente inteligente de segurança local, recebendo as informações de atualização e retornando o status de cada base de dados, contendo a data e a hora da última atualização e um indicativo de como foi feita a atualização (se automaticamente pelo sistema ou por intervenção do administrador de segurança) de cada base de dados (Figura 3.10).

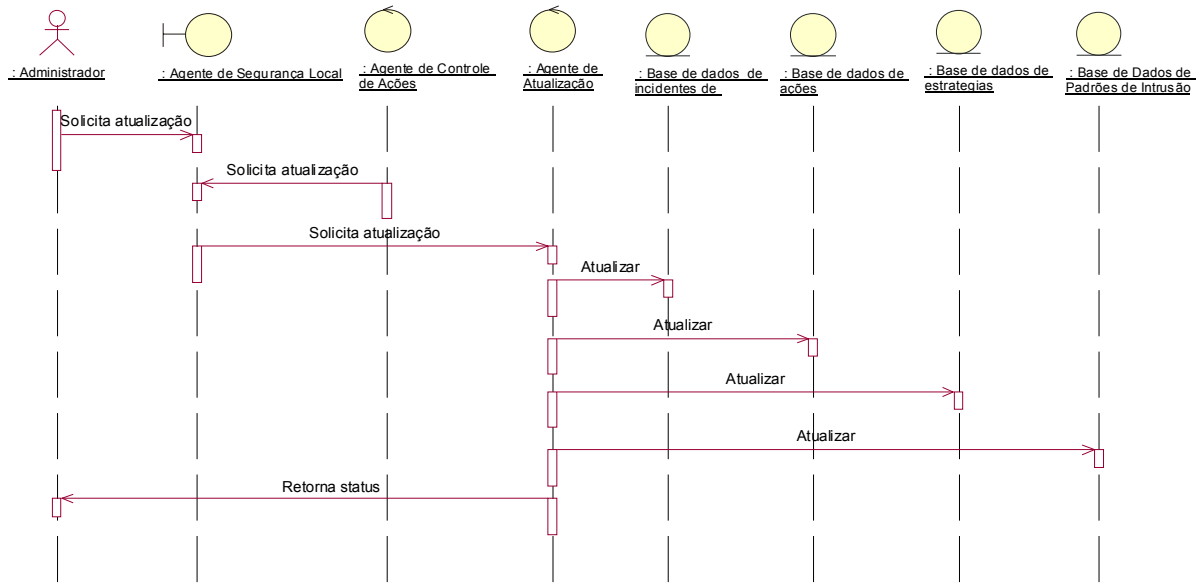


Figura 3.10 – Diagrama de Seqüência do Agente de Atualização (notação UML)

O Agente de Integridade funciona como um pequeno detector de anomalia, que busca por eventos não esperados ou diferentes do normal da sociedade. Caso ocorra alguma anomalia, esse envia os resultados da ação tomada ao agente de segurança local, que, por sua vez, notifica o administrador (Figura 3.11).

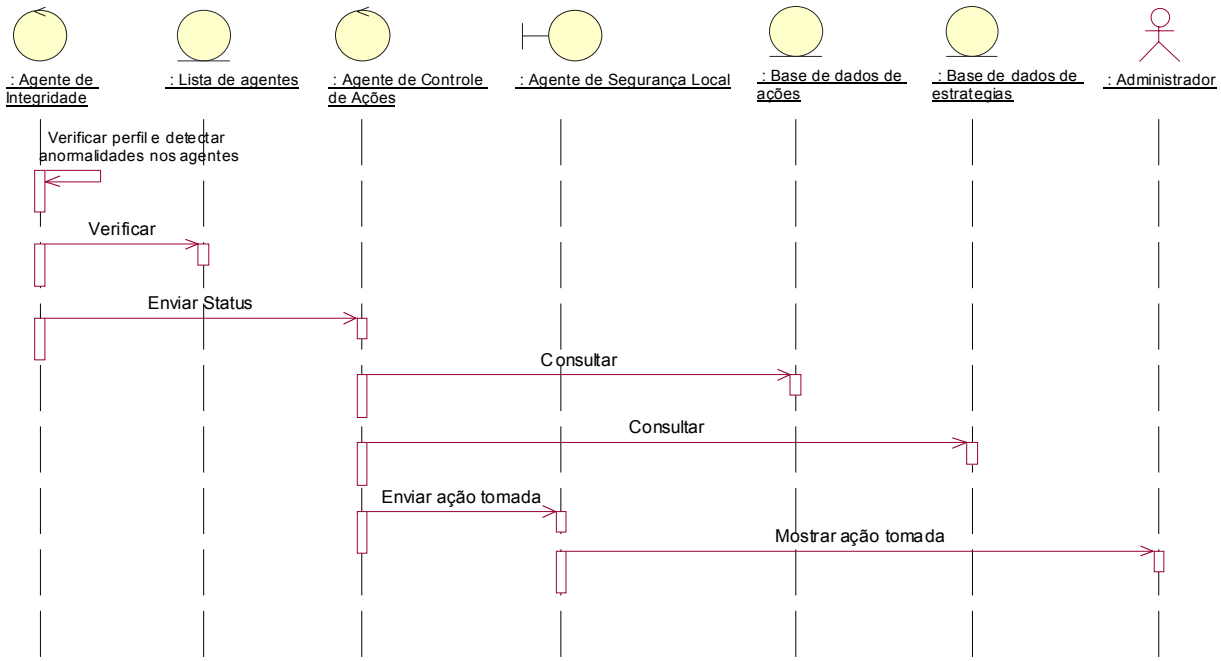


Figura 3.11 – Diagrama de Seqüência do Agente de Integridade (notação UML).

O agente de segurança local gerencia a sociedade de agentes e faz a interface do sistema com o administrador de segurança, permitindo a sua interação com o SDI. É através desse agente que o administrador se comunica com o sistema, enviando e recebendo informações importantes para a administração dos recursos protegidos da organização. Ele está presente na maioria dos cenários apresentados.

3.5 Considerações finais

Neste capítulo foi apresentada uma proposta de um modelo de SDI, formado por uma sociedade de agentes inteligentes, destacando-se a função definida para cada agente e demonstrando a sua interação em sociedade.

Para a implementação do protótipo, foi escolhida a plataforma ZEUS, por ter atendindo os requisitos essenciais ao desenvolvimento do sistema de detecção de intrusão proposto neste trabalho (Apêndice A).

4 IMPLEMENTAÇÃO DOS AGENTES SENSORES DE REDE E DE *HOST*

Neste capítulo, é apresentada a operacionalização dos agentes sensores de rede e de *host*, destacando-se os aspectos relacionados ao projeto e implementação do protótipo proposto, bem como a utilização da plataforma ZEUS de construção de agentes.

4.1 Considerações iniciais

A implementação dos agentes sensores, responsáveis por coletar os dados que serão utilizados na análise pelos agentes responsáveis por identificar as tentativas de intrusão, é o ponto de partida para a construção do sistema proposto. Para efeito de ilustração, optou-se em implementar um agente sensor de rede, com a capacidade de trabalhar nas plataformas Unix e Windows NT e um agente sensor de host, com a tarefa de coletar dados em sistemas de log, baseados na plataforma Windows.

Para a implementação dos agentes sensores, foi utilizada a linguagem Java 1.2.2 (Java, 2000), a ferramenta ZEUS versão 1.3b (Zeus, 2000) e, especificamente no programa externo do agente sensor de *host*, foi utilizada a linguagem C (NAVIA, 2001).

Implementou-se, também, parte do agente de monitoramento, para tornar possível a comunicação inter-agentes. Esse agente será responsável por receber os dados coletados pelos agentes sensores de rede e de *host* e armazená-los (em uma interface gráfica) para posterior análise. Porém, não será dada ênfase à implementação de seu programa externo, em virtude de ser tema de pesquisa em desenvolvimento por outra dissertação (DIAS, 2001).

O processo de construção da sociedade de agentes é baseado na metodologia de desenvolvimento de agentes ZEUS (COLLINS; NDUMU, 1999c). O processo apresenta-se em uma série de fases de desenvolvimento e consiste em atividades globais e individuais que implementam aspectos particulares de um agente (COLLINS; NDUMU, 1999a). Algumas fases, descritas a seguir, são derivadas dos níveis de abstração discutidos no Apêndice A.

i) Criação da Ontologia;

ii) Criação dos Agentes;

iii) Configuração dos Agentes Utilitários;

iv) Configuração do Agente Tarefa;

v) Implementação do Agente.

4.2 Criando a ontologia

Antes de qualquer agente ser implementado, foi definida a ontologia da aplicação, que são conhecimentos declarativos que representam os conceitos significativos, atributos e valores dentro do domínio da aplicação.

Para efeito de teste do protótipo, foram criados fatos do tipo **Saída**, **Pacotes** e **Eventos**, que são relacionados aos agentes pertencentes à sociedade, ou seja, o agente de monitoramento produzirá **Saída**, o agente sensor de rede produzirá **Pacotes** e o agente sensor de *host* produzirá **Eventos**. O uso da ontologia será mostrado nas seções seguintes.

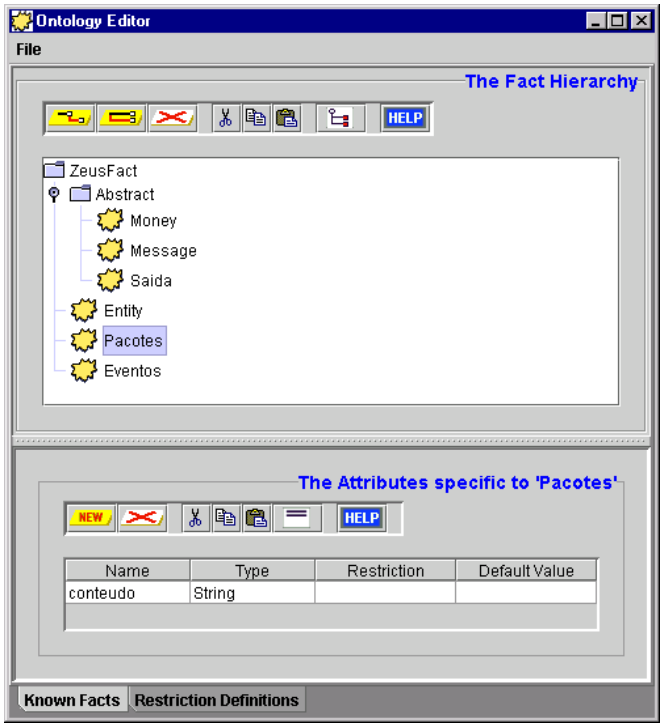


Figura 4.1 - Criando a ontologia

À proporção que o sistema for sendo ampliado, a ontologia (Figura 4.1) poderá ser incrementada, de acordo com as necessidades, sem causar impactos indesejáveis no sistema.

4.3 Criando os agentes

Durante essa fase, um agente ZEUS genérico foi configurado (podendo ser reconfigurado posteriormente com as responsabilidades específicas para a aplicação a que se destina), tranformando-se em agente tarefa. Esse processo envolveu, dependendo da natureza do agente, até quatro sub-fases:

- a definição do agente - onde foram especificadas suas tarefas, recursos iniciais e planejamento de habilidades;
- a descrição das tarefas - onde foram especificadas a aplicabilidade e atributos das atividades do agente;

- a organização do agente - onde o contexto social de cada agente é especificado;

- a coordenação do agente - onde cada agente foi equipado com as habilidades sociais para interação.

Na Figura 4.2, verifica-se que foram criados três agentes tarefas: o agente sensor de rede (NetAgent), o agente sensor de *host* (*Host*Agent) e o agente de monitoramento (SMA). A cada agente foi associada uma tarefa, sendo elas a **CapturarPacote**, **CapturaEventos** e a **EscreverSaída**, respectivamente.

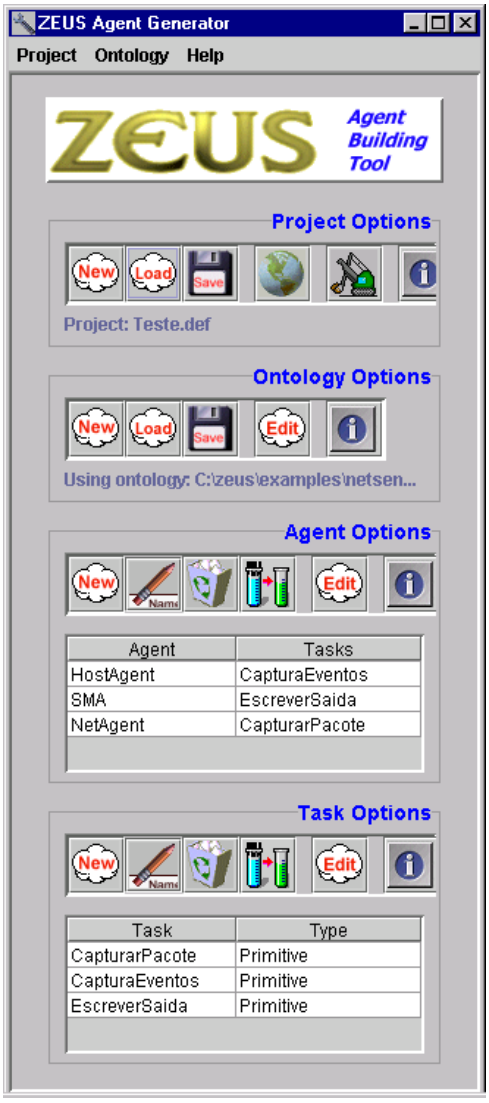


Figura 4.2 - Criando agentes tarefas

As tarefas dos agentes são configuradas conforme as pré-condições e pós-condições estabelecidas na modelagem de cada agente (Apêndice B). Para efeito de testes, essas configurações foram simplificadas nos agentes sensores (Figura 4.3) e modificadas para o agente de monitoramento (Figura 4.4).

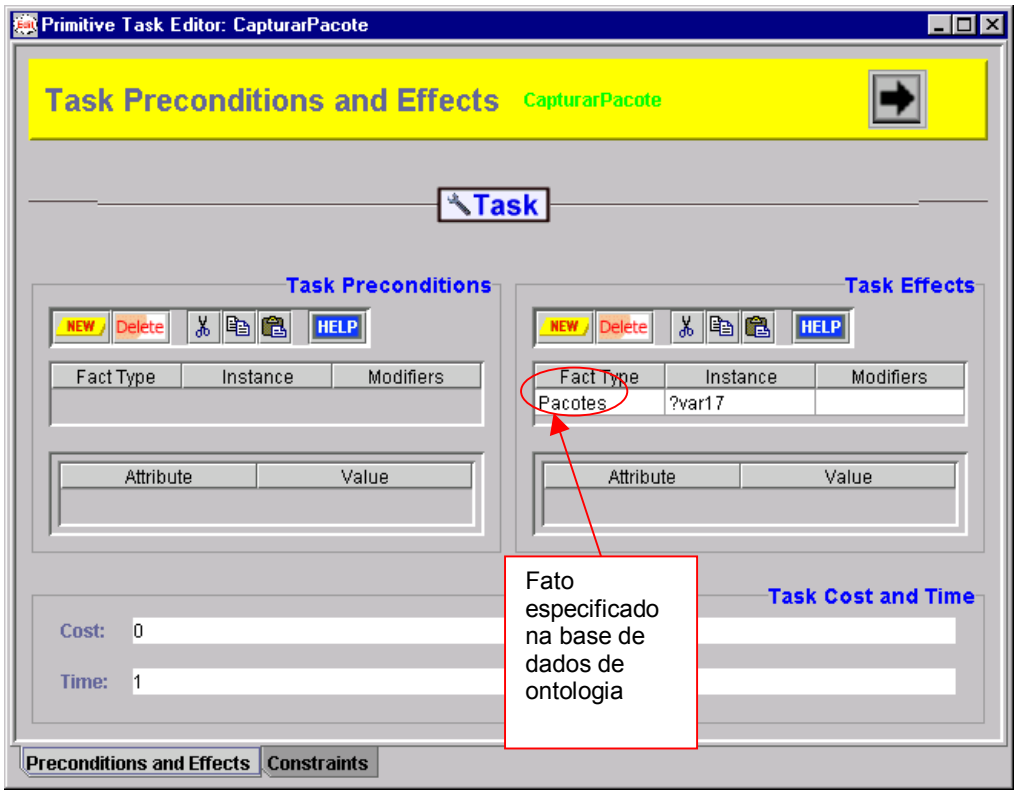


Figura 4.3 – Configurando a tarefa do agente sensor de rede

Conforme foi mencionado anteriormente, o agente sensor de rede tem como saída **Pacotes** e o agente sensor de host tem como saída **Eventos**. Portanto, na tarefa **CapturarPacote**, adicionou-se no campo *Task Effects* o fato do tipo **Pacotes**, conforme Figura 4.3 e, na tarefa **CapturaEventos**, adicionou-se o fato do tipo **Eventos**. Já no agente de monitoramento (SMA), optou-se em utilizar como pré-condições as saídas dos agentes sensores de *host* e de rede, ou seja, a tarefa **EscreverSaída** tem no campo *Task Preconditions* os fatos **Pacotes** e **Eventos** e no campo *Task Effects* tem-se o fato do tipo **Saída** (Figura 4.4).

Em resumo, o agente de monitoramento só poderá **EscreverSaída** se o mesmo receber as informações coletadas pelos dois agentes sensores.

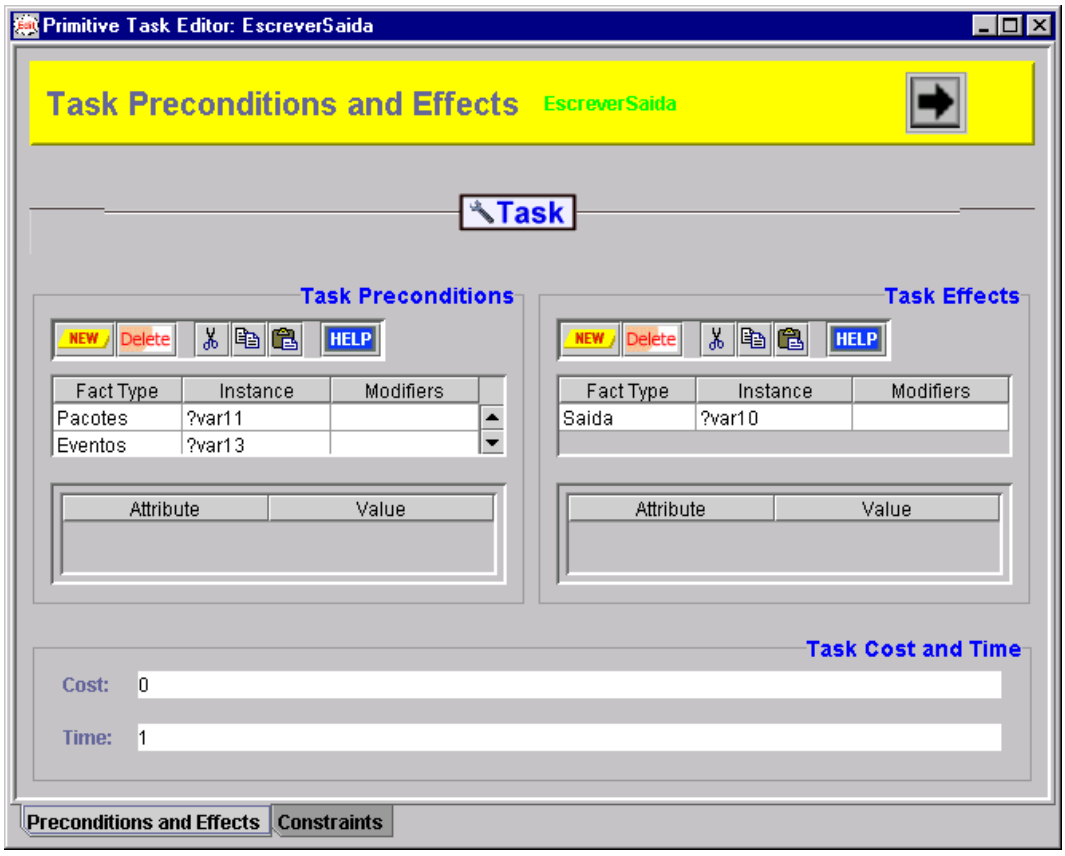


Figura 4.4 – Configurando a tarefa do agente de monitoramento

Por “default”, na arquitetura ZEUS, os agentes não conhecem os nomes e habilidades dos outros da mesma sociedade, assim se um agente precisa do serviço de outro, precisará contactar um serviço de diretório para descobri-lo. Como opção, eles podem ter conhecimento pré-existente de outros agentes, especialmente se eles interagem com freqüência. Há quatro tipos diferentes de relações que podem ser definidos entre agentes:

- superior - possui autoridade maior que outros agentes, podendo emitir ordens, que deverão ser obedecidas;
- subordinado – tem menos autoridade que o superior e tem que obedecer suas ordens;
- *co-worker* – fazem parte da mesma sociedade e será consultado quando qualquer recurso for requerido (antes do relacionamento *peer*);

- *peer* – relacionamento default sem suposição sobre a interação do agente.

Portanto, na etapa de organização do agente, definiu-se o relacionamento de NetAgent e *Host*Agent como “subordinados” em relação ao SMA e associou-lhes às suas habilidades, conforme mostrado na Figura 4.5.

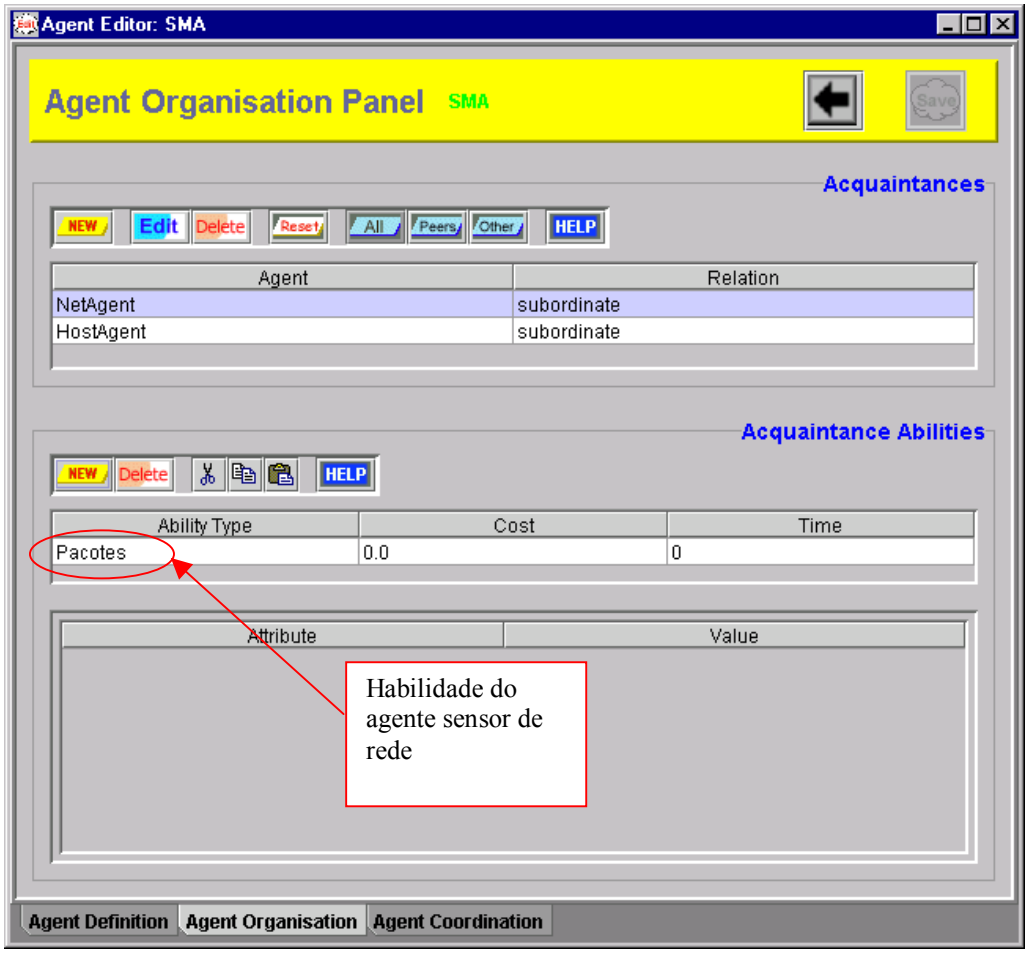


Figura 4.5: Organização dos agentes sensores

Na etapa de coordenação dos agentes, os mesmos foram equipados com o protocolo de coordenação e estratégias, que implementam vários aspectos da conversação tipo rede de contrato (contract-net), discutidos no Apêndice A. Dessa forma, definiu-se que NetAgent e *Host*Agent seriam *Participantes* e o SMA foi configurado para ser *Iniciador* e *Participante* simultaneamente, conforme mostrado na Figura 4.6

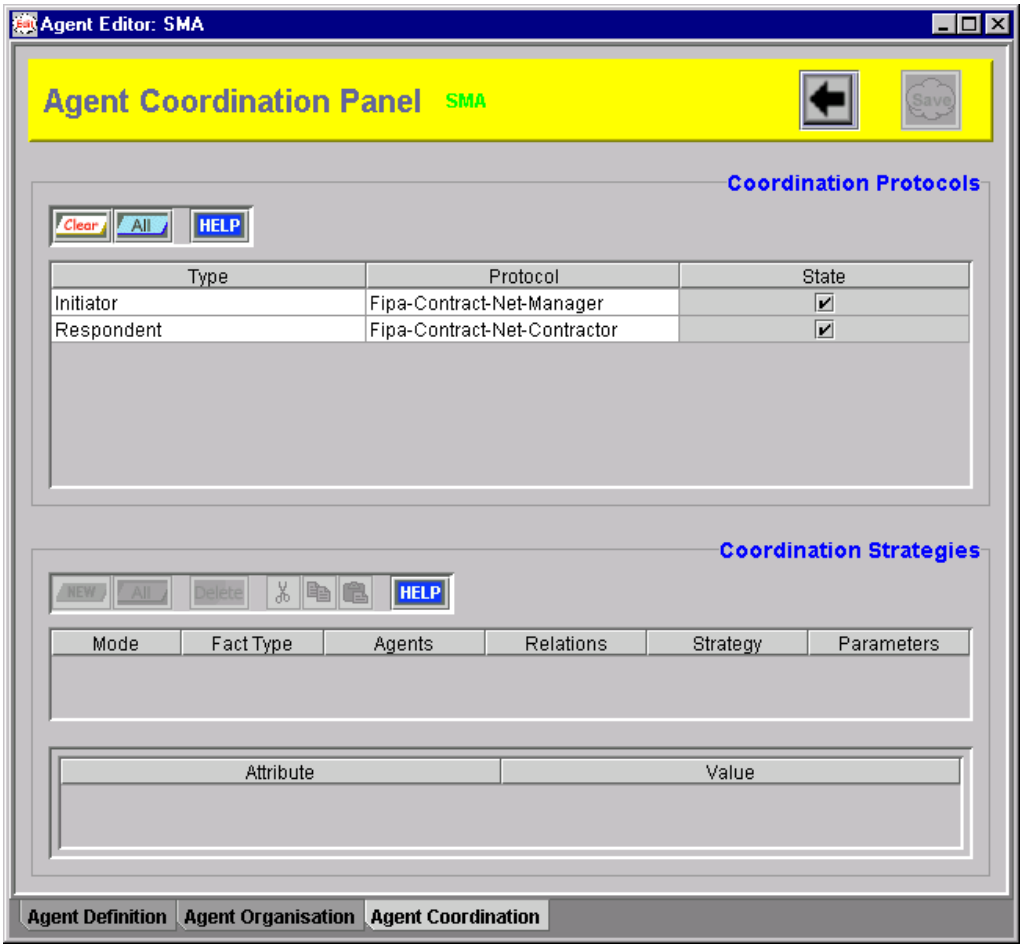


Figura 4.6: Equipando o agente SMA com o protocolo de coordenação

Concluindo-se a fase de criação dos agentes, verifica-se que os mesmos estão preparados para trabalhar em sociedade. Porém, deve-se ressaltar o fato de que apesar de se ter definido uma tarefa específica para cada agente, eles ainda não são capazes de realizá-las, pois tais tarefas serão realizadas por um programa externo.

4.4 Configurando os agentes utilitários

Nessa fase, foram definidos os atributos dos agentes utilitários (Servidor de Nomes, Facilitador e Visualisador) que fornecem a infra-estrutura de apoio para a sociedade de agente.

Um sistema multiagentes deve ter pelo menos um Agente Servidor de Nomes (ANS), porém não é obrigatório o uso do Facilitador e do Visualisador. Entretanto, podem ser incluídos dependendo da natureza da aplicação e em situações onde a aplicação precisa ser monitorada ou analisada, o uso do Visualisador se faz necessário. Os ANS´s mantêm um registro dos agentes, habilitando-os a mapear identidades de outros e sua localização de rede lógica. Isto é necessário porque mesmo que os agentes conheçam os nomes dos demais, desconhecem as suas localizações.

A Figura 4.7 mostra a tela de configuração dos agentes utilitários. Nessa fase pode-se renomear e especificar os endereços físicos onde tais agentes irão executar.

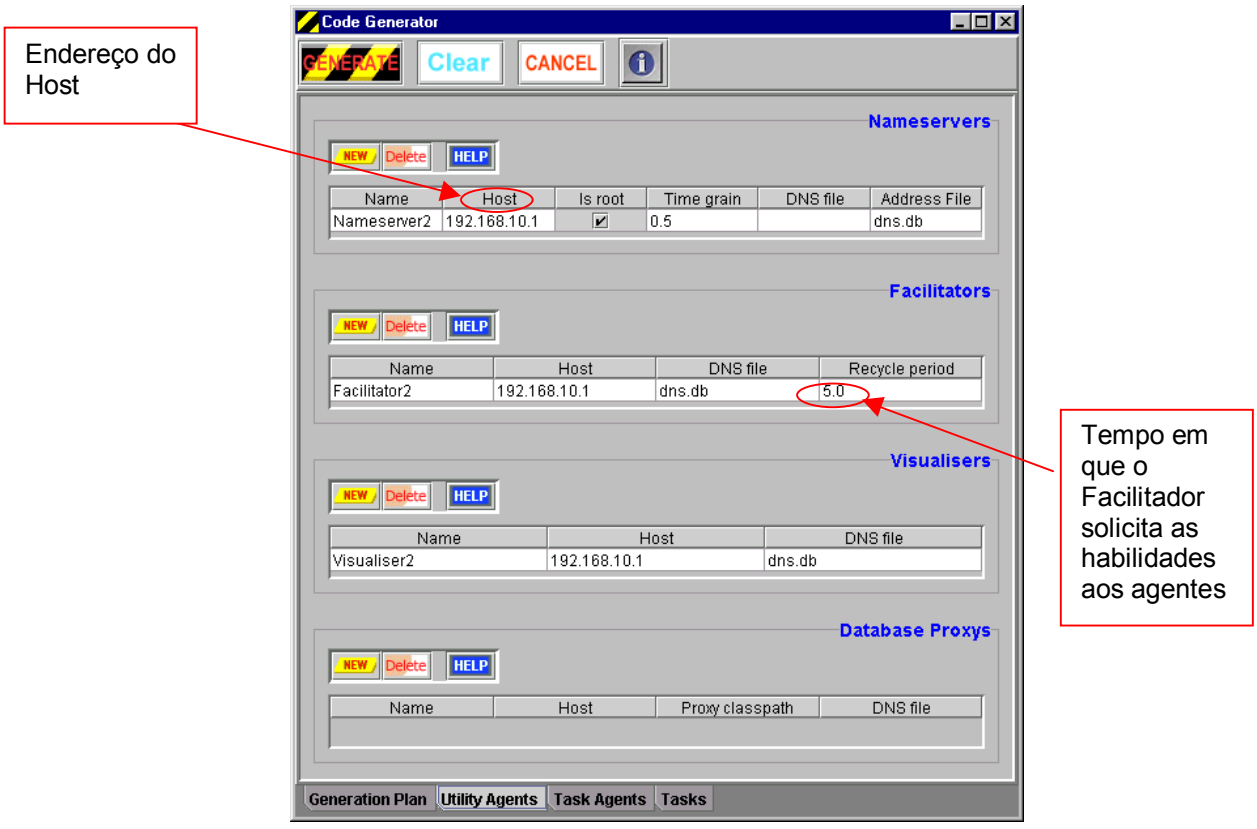


Figura 4.7: Configurando os agentes utilitários

4.5 Configurando os agentes tarefas

Nessa fase, os parâmetros de execução dos agentes tarefas foram especificados (Figura 4.8). Isto envolveu informar em quais máquinas os agentes irão executar (*host*) e quais recursos externos (*Database Extension*) e programas (*External Program*) eles irão acionar. Os programas externos serão detalhados nas Seções 4.7 e 4.8.

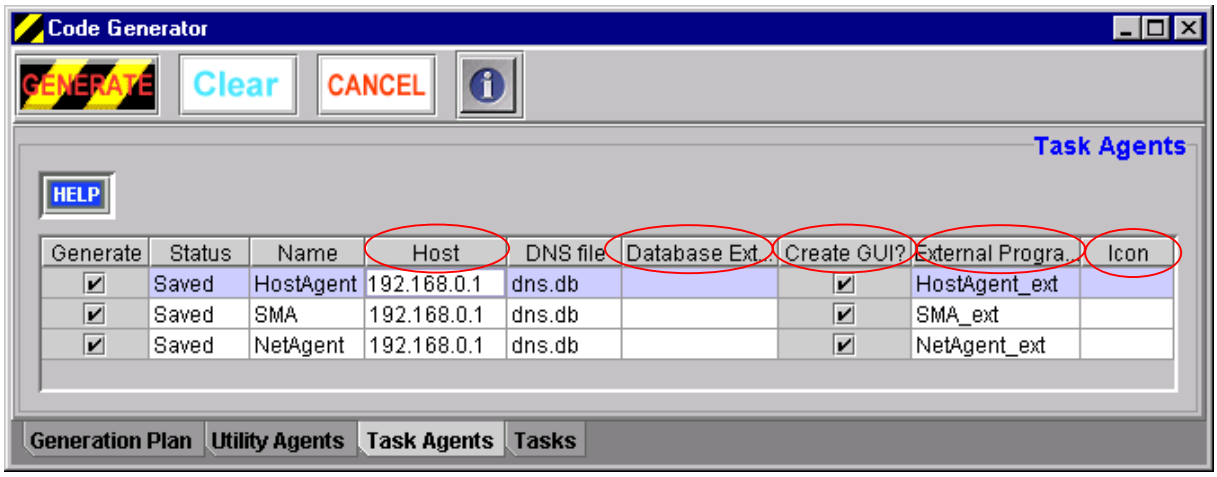


Figura 4.8: Configurando os agentes tarefas

Nesse nível de configuração é permitido optar em criar interfaces gráficas para cada agente (campo *Create GUI?*) e associá-los a um ícone (campo *Icon*). A interface gráfica permite o acesso a todos os componentes do agente, tais como Caixa Postal, Bases de Conhecimento, Máquina de Coordenação etc, descritas no Apêndice A.

4.6 Gerando os agentes

Nessa fase, o *Gerador de Código* (Collins e Ndumu, 1999a) pôde ser invocado e o código fonte do agente foi gerado automaticamente (Figura 4.9). Com isso, foram geradas as classes *Host*Agent, *NetAgent*, *SMA*, *CapturarPacote*, *CapturaEventos* e *EscreverSaída* em Java, prontas para serem compiladas.

Isto permitiu uma concentração maior no trabalho de realizar a implementação das tarefas específicas para a aplicação, recursos externos, programas externos (por exemplo, uma interface de usuário) e estratégias de interação.

Os programas externos dos agentes tarefas utilizam os métodos públicos da API Zeus (implementando a interface *zeus.agents.ZeusExternal*), sendo, tipicamente, uma interface que recebe instruções e envia os resultados.

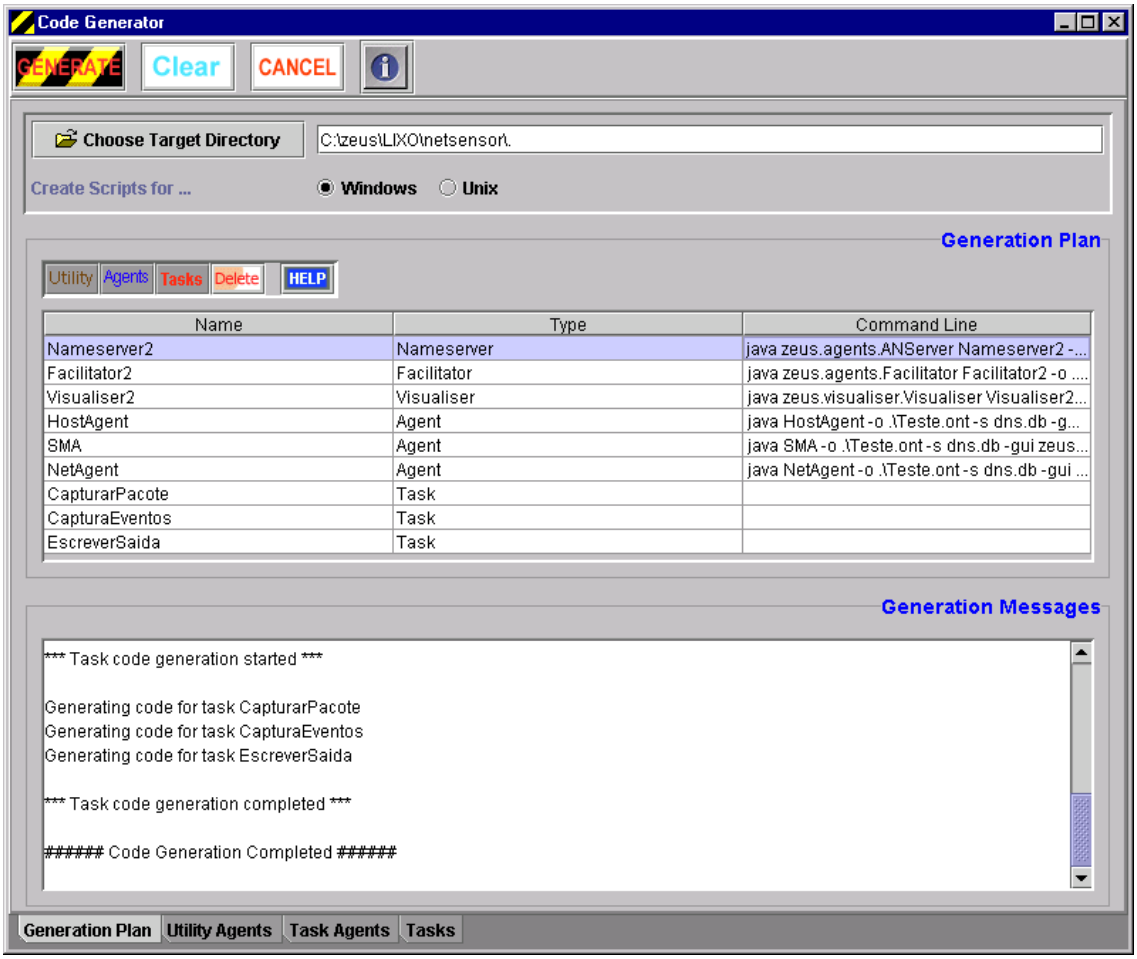


Figura 4.9: Gerando os agentes

4.7 Implementação do programa externo do agente sensor de rede

O agente sensor de rede é componente fundamental no processo de aquisição de dados do sistema de detecção de intrusão proposto. Ele foi projetado para trabalhar em várias plataformas, tendo sido testado nos ambientes Windows 98/NT/2000 Server e Workstation. Porém, como ele foi todo desenvolvido utilizando-se a Linguagem Java, a sua portabilidade para outras plataformas é possível.

Sua principal tarefa é capturar os pacotes que estão trafegando na rede e disponibilizá-los para análise. Com isso, diversos tipos de ataques podem ser identificados e prevenidos. Esse agente interage com a rede de forma passiva, somente "escutando" seu tráfego como um *sniffer*. À medida que os pacotes vão sendo capturados, eles são enviados para os agentes apropriados para que possam ser analisados.

O agente sensor de rede ativa um programa externo, que interage com o segmento de rede para efetuar a captura dos pacotes. Para isso, ele faz uso das bibliotecas Jpcap (Fujii, 2000) e libpcap (Winpcap, 2000). Seu uso foi requerido devido a grande complexidade técnica envolvida na captura dos datagramas[26].

A biblioteca de captura de pacotes libpcap interage junto ao kernel do sistema operacional para fornecer um acesso rápido a uma cópia dos pacotes para análise.

O uso dessa biblioteca apresenta diversas vantagens, entre as quais: isola os aplicativos dos detalhes da tecnologia de rede utilizada (Ethernet, FDDI, SLIP, etc); ajuda a manter a portabilidade dos programas com relação à plataforma, uma vez que existem versões para diversos sistemas operacionais; e dá acesso a um esquema de filtragem de pacotes bastante poderoso e flexível.

---

[26] Datagrama é "*uma entidade de dados, auto-contida, que possui suficiente informação para ser conduzida do computador de origem ao destino, sem depender nem das trocas anteriores entre aqueles pontos nem da rede transportadora"* (Internet RC1594). *DATAGRAMAS* ou pacotes são unidades de mensagens tratadas pelo Protocolo Internet (IP) e transportadas por ela.

A Jpcap é um conjunto de classes Java que permite a programação da biblioteca *libpcap* em um nível de abstração maior, possibilitando alto grau de produtividade. Esse pacote usa a *libpcap* e a *Raw Socket API* para permitir a captura e envio de pacotes IP (IPV4 e IPV6) de aplicações Java.

No Apêndice C, a arquitetura padrão de comunicação da Internet é brevemente discutida. Isto é importante devido às informações transmitidas sob esse protocolo serem o objeto de estudo do agente sensor de rede.

4.7.1 Captura e filtragem dos pacotes da rede

A captura de pacotes da rede é uma tarefa bastante delicada e deve ser feita com muito critério. Coletar todos os datagramas, apesar de ser a situação ideal, não é tecnicamente viável. Todavia, usar uma filtragem muito permissiva pode comprometer a análise de segurança do sistema. Portanto, é preciso considerar a sobrecarga gerada e o grau de hostilidade do ambiente para determinar-se um *threshold* satisfatório.

O uso desses filtros é muito importante, pois alguns ataques são baseados na exploração de vulnerabilidades bastante conhecidas. A exemplo disso, tem-se o *IP Spoofing*, que consiste em informar o número IP da máquina de forma incorreta, geralmente trocando-o por um endereço qualquer, através de manipulação direta dos campos do cabeçalho do datagrama. Assim, quando um *host* A quer se conectar a um *host* B, a identificação é feita através do campo ENDEREÇO DE ORIGEM do pacote. Se o IP do cabeçalho enviado pela máquina A for falso (for o IP do computador C, por exemplo), o *host* B, na falta de outra forma de identificação, acredita estar se comunicando com C.

Com o uso dessa técnica, o invasor pode ter acesso a máquinas que confiam no IP que foi falsificado. Isso traz sérias implicações nos serviços "r" do UNIX, tais como o *rlogin* e o *rshell*, que são baseados na noção de autenticação através de números IP de máquinas seguras. Um problema clássico desse tipo ocorre quando um atacante envia pacotes UDP para um servidor de NFS (*Network File System*) como se esses viessem de uma máquina cliente, autorizada a acessar

completamente algum diretório. Assim, o servidor entende que trata-se de uma operação confiável e garante direitos normais a um usuário ilegítimo.

Essa técnica é geralmente associada a outros tipos de ataques, como o SYN Flood, um dos mais populares tipos de negação de serviço (DoS). Através dele, consegue-se inutilizar qualquer serviço baseado em TCP. O ataque consiste basicamente em enviar um grande número de pacotes de abertura de conexão (*flags* SYN ligado e ACK desligado), com um endereço de origem falso, possivelmente inexistente, para um determinado servidor.

O servidor, ao receber essas requisições, passa a alocar recursos para dar início à comunicação. Depois, ele envia um datagrama de resposta (SYN e ACK ligados) e fica esperando uma confirmação da máquina cliente, que nunca chegará, pois o endereço original é inatingível. Em um dado momento, a máquina não consegue mais dispor de recursos e todos os pedidos de abertura de conexão são descartados. A partir daí, os usuários legítimos do sistema são privados da utilização dos serviços.

Com o processo de filtragem de eventos, é possível implementar vários agentes sensores com habilidades distintas, permitindo maior flexibilidade ao sistema. A título de exemplo para a implementação de um protótipo do agente sensor de rede, foram utilizados alguns filtros de acordo com os diversos tipos de ataques conhecidos:

Filtro: *(tcp and (tcp[13] & 2 != 0) and dst net 192.168.10*

Descrição: o primeiro "tcp" da regra indica que se deseja capturar somente os pacotes do tipo 6 (TCP). O " tcp[13]" refere-se ao décimo terceiro byte do cabeçalho do segmento, onde estão localizados os bits de código. O "2 != 0" é a notação utilizada para se dizer que só os pacotes com o flag SYN ligado devem ser considerados. O "dst net 192.168.10" identifica os datagramas com destino à rede 192.168.10. Dessa forma, o agente sensor de rede passa a capturar somente os pacotes TCP de requisição de conexão destinados a uma rede específica.

Utiliza-se a regra ***tcp and (tcp[13] & 3 != 0)*** para capturar pacotes de conexões (SYN) ou desconexões (FIN). A regra ***tcp and (tcp[13] & 7 != 0)*** habilita o agente sensor de rede a capturar pacotes com os flags SYN, FIN ou RST. Dessa forma, consegue-se delimitar e conhecer a data e hora do início (SYN) e fim (FIN ou RST) de cada conexão TCP e extrair informações adicionais, tais como: tempo de duração, *hosts* participantes, portas envolvidas e o número de bytes enviados em cada direção, para serem analisados posteriormente.

Filtro: *port finger or port ftp or dst port smpt or tcp port 113 or port telnet or port login or port 111*

Descrição: com essa regra, o agente passa a aceitar todo pacote TCP com porta de origem ou destino 79 (finger), 21 (ftp), 113 (ident), 23 (telnet), 513 (rlogin), pacotes com destino à porta de e-mail smtp (25) e qualquer pacote TCP ou UDP com porta 111 (portmapper) de origem ou destino.

Com o serviço de telnet, o usuário pode entrar em uma máquina remota, através da rede, e trabalha nela como se estivesse sentado num terminal diretamente conectado a ela. Essa porta é bastante utilizada pelos *hackers* em seus ataques e, por isso, tem merecido maior atenção.

Filtro: *icmp[0] = 8 and icmp[0] = 0*

Descrição: através desse filtro, pode-se detectar os pacotes ICMP com as mensagens de "Requisição de eco" e "Resposta de eco" respectivamente. Assim, tem-se um meio de tentar evitar diversos tipos de ataques que usam esse protocolo de controle, dentre os quais o SMURF. Nesse tipo de ataque, o *hacker,* envia pacotes ICMP com um pedido de requisição de echo para vários *hosts* da rede, contendo o endereço da máquina a ser atacada no campo ENDEREÇO DE ORIGEM do cabeçalho dos datagramas. Ao receberem os pacotes, os *hosts* têm sua resposta direcionada a um *host* específico, causando um "ICMP *flooding*" na máquina alvo. Esse ataque afeta os seguintes sistemas operacionais: Windows 95, Windows NT Server com Service Pack 4, Linux e FreeBSD, causando variações no desempenho durante o ataque.

Filtro: *ip and ip[19] = 0xff*

Descrição: esse filtro é responsável por capturar todos os pacotes IP de uma difusão. Isso pode ser interessante, pois diversos ataques de negação de serviço são baseados no recurso de *broadcast* do protocolo IP, inclusive aqueles que utilizam o ICMP.

Filtro: *ip and ip[12:4] = ip[16:4]*

Descrição: com essa simples regra, o agente sensor de rede consegue capturar pacotes que possuem os endereços de origem e destino iguais. Essa é a base do ataque "*Land*", que pode colocar um sistema em loop infinito. Algumas versões de sistemas operacionais, como o *Windows* 95 e *Windows* NT *Workstation* com *Service Pack* 3 têm suas máquinas travadas quando sofrem esse tipo de ataque pela porta 139 e as distribuições mais antigas do Linux apresentam queda de desempenho quando várias cópias desse ataque são disparadas contra ele.

Filtro: *udp port 161 or udp port 162*

Descrição: com esse filtro o agente monitora as portas utilizadas pelo serviço de SNMP (*Simple Network Management Protocol*). Inicialmente, o SNMP foi desenvolvido para monitorar *hosts*, roteadores e redes, em busca de algum tipo de problema. Entretanto, os *hackers* têm usado esse protocolo para obter informações preciosas sobre o sistema, que podem ser usadas em vários tipos de ataques.

Filtro: *ip and udp port 2049*

Descrição: a porta UDP 2049 é utilizada pelo serviço de NFS. Esse aplicativo permite que um conjunto de computadores acessem sistemas de arquivos remotos como se fossem locais. Através do IP *Spoofing* é possível dissimular os esquemas de segurança e garantir acesso a arquivos e pastas protegidos.

Filtro: *tcp and port 6667*

Descrição: esse filtro reconhece o tráfego de informações trocadas entre os programas de IRC (*Internet Chat Relay*). Isso pode ser importante porque mesmo que uma empresa proíba o uso de tais aplicativos, *hackers* que consigam invadir a

rede podem instalar um servidor de IRC para comunicarem-se com outros intrusos. Assim, é interessante usar esse tipo de filtragem para monitorar possíveis máquinas comprometidas.

Filtro: *ip[6:2] & 0x2000 != 0*

Descrição: esse filtro é utilizado para detectar a ocorrência de fragmentos de pacotes. Isso é importante para detectar ataques relativos à fragmentação. Um ataque desse tipo faz uso da fragmentação de datagramas para enviar dados maliciosos a uma rede de computadores. Para isso, ele usa de engenhosidade para conseguir burlar as regras de alguns tipos de filtros de pacotes, utilizados por roteadores e *firewalls*.

Essa técnica parte da premissa de que a filtragem de pacotes fragmentados ocorre somente para o primeiro fragmento da série, pois se esse não conseguir passar pelo mecanismo de segurança, a conexão com o *host* destino não será efetuada. Então, é feita uma modificação no pacote de modo que o primeiro fragmento não contenha a solicitação de conexão, mas o segundo sim.

Como o filtro de pacotes do firewall só verifica o pacote inicial em busca do *flag* SYN ligado, os pacotes são todos transmitidos com sucesso, permitindo que o datagrama indesejável seja remontado na máquina de destino.

4.7.2 Detalhes da implementação

Aqui serão discutidas algumas técnicas e soluções adotadas na implementação das diversas partes do programa externo do agente sensor de rede. Para um melhor entendimento, serão utilizados diagramas de estado da notação UML e trechos de código comentados.

As três classes principais escritas para o programa são mostradas na Figura 4.10. Primeiramente, um objeto da classe *NetAgent_ext* é responsável pelas inicializações e ativação da captura de pacotes, que será realizada pela classe *CapturaPacotes*[27]. À medida que os datagramas vão sendo obtidos, eles são tratados, formatados e passados para um objeto da classe *EnviaPacotes*, responsável pelo envio das informações selecionadas ao agente de monitoramento (SMA) para serem processadas.

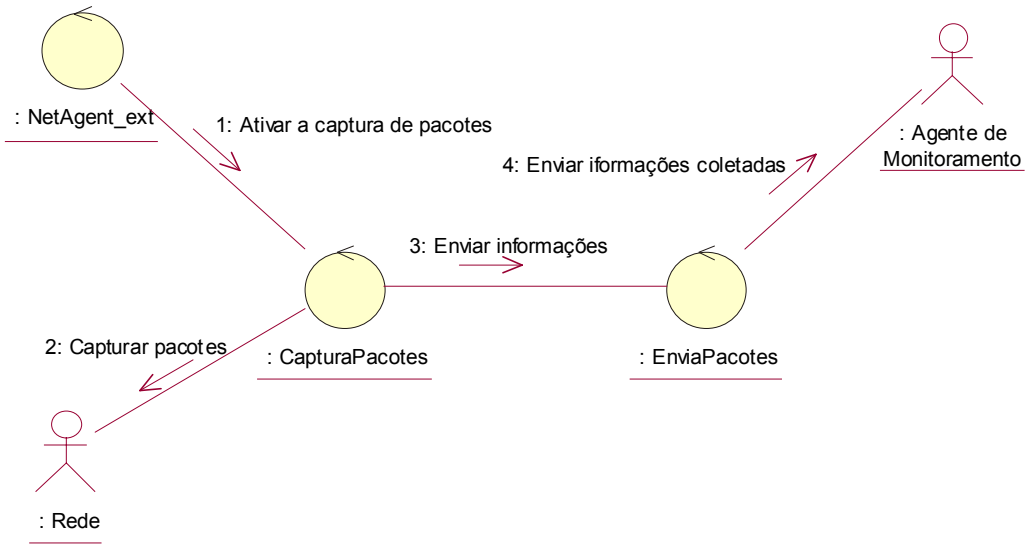


Figura 4.10 - Diagrama de colaboração do programa externo do agente sensor de rede

O diagrama de estado da classe *CapturaPacotes* é apresentado na Figura 4.11, que será tomado como base para serem fornecidas maiores informações sobre a implementação.

[27] Esta não é a classe (*CapturarPacote*) que foi gerada automaticamente pela ferramenta ZEUS.

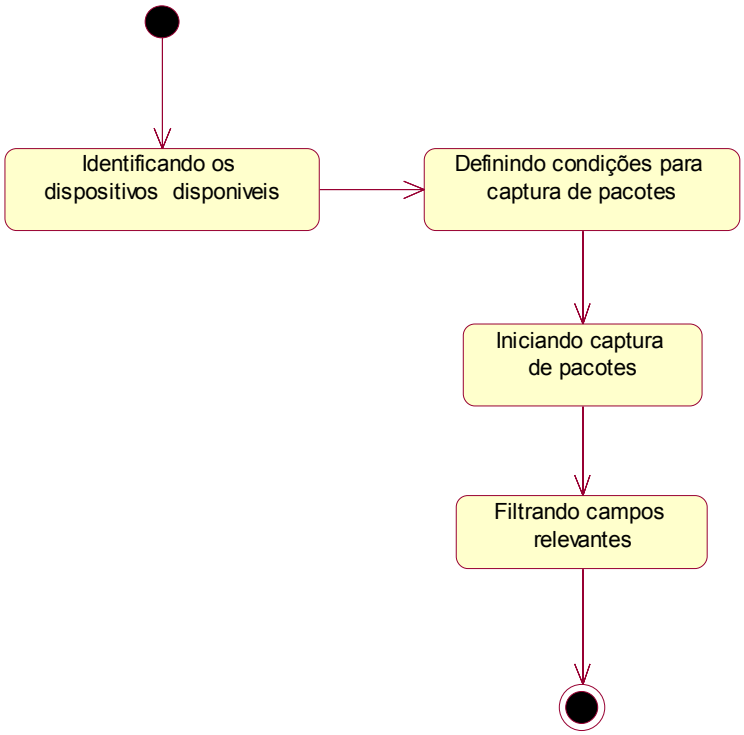


Figura 4.11 - Diagrama de estado da classe *CapturaPacotes*

Inicialmente, uma instância da classe Jpcap é criada, habilitando a captura de pacotes através da rede (Figura 4.12) O construtor utilizado para essa classe requer diversos parâmetros de configuração, dentre os quais os mais importantes são:

1. o nome da interface utilizada;
2. o número máximo de bytes capturados por vez;
3. o modo de interação com a rede;
4. a filtragem utilizada.

```
//inicializa vetor com as interfaces encontradas
String[] devices=Jpcap.getDeviceList();
//Só há uma placa de rede
Device = devices[0];
 try
        {
                //chamada do construtor
                jpcap=new Jpcap(device,1500,true,50);
                //filtro depende do que se deseja capturar
                jpcap.setFilter(filtro,true);
        }
 catch(IOException e)
jpcap.loopPacket(-1,this); //inicia captura de pacotes
```

Figura 4.12 - Código exemplo do *CapturaPacotes*

O primeiro parâmetro é obtido através da utilização do método *getDeviceList*( ), que informa as várias interfaces de comunicação (placas de rede ou placas de fax modem) do *host* para que o usuário ou sistema escolha qual a mais adequada para a captura dos datagramas.

O segundo parâmetro foi definido em 1500 *bytes*, que é o número máximo de dados em um quadro *Ethernet*. O modo de interação com a rede é o *promíscuo*, que habilita a interface de comunicação selecionada a capturar todos os pacotes que estão trafegando na rede.

Em seguida, é necessário passar para o objeto Jpcap as definições dos tipos de pacotes que devem ser capturados. Nessa implementação, foram aplicados os filtros apresentados na Seção 4.7.1.

Por fim, o método *loopPacket* de Jpcap é chamado e a captura dos pacotes é iniciada. A cada datagrama copiado, são extraídas informações relevantes, visando simplificar o processo de análise pelos agentes do SDI.

Esse processo consiste em selecionar e formatar informações importantes que serão repassadas para a classe *EnviaPacotes*, tais como: a hora (em milisegundos) do recebimento do pacote, os endereços IP de origem e destino, as portas de origem e destino (TCP e UDP), os campos IDENTIFICAÇÂO, DF (DO NOT FRAGMENT) e MF (MORE FRAGMENTS) do pacote IP, os BITS DE CÓDIGO (TCP) e a mensagem do cabeçalho ICMP.

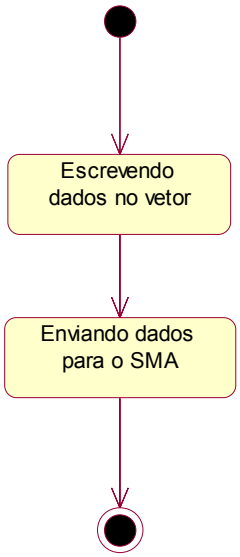


Figura 4.13 - Diagrama de estado da classe *EnviaPacotes*

As informações recebidas pela classe *EnviaPacotes* são armazenadas em um vetor e enviadas ao agente SMA (Figura 4.13) após atingirem um número pré-definido de pacotes capturados. Esse procedimento é necessário para aumentar o desempenho do envio de mensagem inter-agentes.

4.8 Implementação do programa externo do agente sensor de *host*

Os agentes sensores de *host* trabalham coletando informações em tempo real de um sistema em particular e disponibilizando-os para análise. Para tanto, eles geralmente interagem com o mecanismo de auditoria do sistema operacional e os *logs* do sistema.

É certo que os *logs* de auditoria gerados pelos sistemas operacionais, aplicações, rede, atividades dos usuários, entre outros, trazem inúmeras informações que podem corresponder a tentativas de invasão. Porém, transformar a enorme quantidade de dados em informações úteis não é uma tarefa trivial. Como os *logs* registrados em cada um dos sistemas - sejam eles *firewalls*, roteadores, servidores UNIX ou NT - contêm, em geral, informações e formatos distintos, nem sempre são compatíveis entre si.

Portanto, os agentes sensores de *host* são projetados para trabalharem em ambientes específicos, sendo necessário um conhecimento mais aprofundado do modo com que cada sistema trata os dados auditados, contrariando o fato de se implementar agentes independentes de plataforma. Para fins de teste, o agente sensor de *host* foi implementado para trabalhar em ambientes *Windows* NT (*Server e Workstation*) e o *Windows* 2000. Esse ambiente foi o preferido devido a diversos fatores: é o ambiente que possui maior número de incidentes registrados (ALLDAS, 2002); disponibilidade de recursos nos laboratórios da UFMA; o fácil acesso à documentação sobre o assunto e a familiaridade com o ambiente.

De fato, o que está sendo referenciado, no momento, não é o agente propriamente dito e sim o núcleo de seu programa externo, que é a parte específica da implementação que realmente interage com o sistema operacional. O restante das classes desenvolvidas em Java poderão perfeitamente ser reutilizadas em implementações futuras de outros agentes sensores de *host* para diferentes ambientes.

Semelhante ao agente sensor de rede, o agente sensor de *host* ativa um programa externo, que interage diretamente com o sistema operacional em específico. Na seção seguinte, são abordados os aspectos mais relevantes do mecanismo de *log* do *Windows* NT.

4.8.1 Sistema de *log* do *Windows NT*

O *Windows* NT fornece um mecanismo padrão de *log* onde as diversas aplicações, inclusive o próprio sistema operacional, armazenam informações importantes sobre o comportamento de software e hardware. Essas informações correspondem a eventos significativos que indicam o estado de funcionamento do sistema.

O sistema de *log* do *Windows* NT é constituído por três arquivos de *log*, conforme descrito no registro do sistema operacional:

i) **o *log* de Aplicação**: armazena os eventos relacionados aos diversos programas genéricos do usuário e aos diversos serviços do sistema

ii) **o *log* de Sistema**: armazena os eventos ocorridos nos drivers de dispositivos ou no próprio *kernel* do *Windows*;

iii) **o *log* de Segurança**: registra os eventos relacionados com o sucesso ou falha de auditoria.

Todos esses arquivos possuem a extensão EVT e só são lidos corretamente através do aplicativo *Event Viewer* do *Windows* ou utilizando-se diretamente funções específicas da API do Windows.

Um cuidado especial que deve ser tomado é que nem sempre os sistemas têm como *default* a coleta das informações necessárias à finalidade do projeto e, sem os mecanismos devidamente configurados e implementados, dificilmente será possível determinar se uma tentativa de invasão foi feita e se o invasor foi bem sucedido.

Porém, um fator importante a ser considerado é que armazenar informações nos *logs* consume recursos computacionais, tais como espaço em disco e tempo de processamento. A quantidade de espaço de disco requerida e a sobrecarga gerada dependem da quantidade de informação a ser registrada no *log*. Portanto, é imprescindível usar o *log* somente para salvar eventos relevantes.

De fato, para se auditar um sistema, é preciso encontrar um compromisso entre o nível de segurança almejado, a sobrecarga gerada e a complexidade da análise dos registros de *log*.

O propósito principal da auditoria em computadores individuais é detectar possíveis casos de mau uso ou comportamento suspeito no sistema. Ela pode informar sobre ações que representam possíveis riscos para a segurança do sistema

e identificar a conta sob a qual as ações foram tomadas. Para isso, algumas atividades específicas do sistema operacional podem ser gravadas em local seguro para posterior análise.

O protótipo do agente sensor de *host* do sistema de detecção proposto está configurado para capturar eventos conforme a orientação disponível na dissertação de mestrado de (KREMER,1999). Nesse documento, elaborado pela Marinha Americana, em conformidade com (C2, 1985), estão disponíveis algumas considerações sobre segurança no Windows NT, inclusive o que deve ser auditado. Além do trabalho de Kremer, também foi tomado como base o documento desenvolvido pela empresa de segurança (Centrax,1999).

Alguns eventos recomendados para a auditoria são:

a) as falhas e sucessos de *login/logoff*;

b) sucesso e falha de acesso a objetos e arquivos;

c) falha e sucesso no gerenciamento de usuários e grupos;

d) falha e sucesso nas mudanças da política de segurança;

e) falha e sucesso no desligamento e reinicialização do sistema;

f) certas chaves do registro;

g) os arquivos essenciais do sistema e

h) manipulação dos arquivos de *log*;

Os arquivos considerados para auditagem são mostrados na Tabela 4.1. É importante ressaltar que essas recomendações não podem ser consideradas finais e utilizáveis em todos os casos. Cada organização deve ser avaliada *in loco* para que suas reais necessidades sejam averiguadas, pois decisões mal formuladas podem colocar em risco o desempenho geral do sistema.

Tabela 4.1- Acesso requerido às funções da API

Arquivo/Diretório/Chave de registro	Tipo	Comentários
..exe	ED	Todos os arquivos executáveis
..bat	ED	Todos os arquivos do tipo batch
..com	ED	Todos os arquivos de comando
..dll	EXD	Todas as DLLs
..ini	ED	Todos os arquivos de inicialização
*.\bootsect.dos	ED	Arquivos do sistema operacional
*.\ntldr	ED	Arquivos do sistema operacional
*.\ntboot.sec	ED	Arquivos do sistema operacional
.\winnt.exe	ED	Arquivos do sistema operacional
.\winnt.com	ED	Arquivos do sistema operacional
.\winnt.bat	ED	Arquivos do sistema operacional
.\winnt.dll	ED	Arquivos do sistema operacional
.\winnt.sys	ED	Arquivos do sistema operacional
.\winnt.ini	ED	Arquivos do sistema operacional, exceto \winnt\profiles*.ini
.\winnt\system32\config.evt	LED	Os arquivos de log
*.\winnt\repair\	ED	Diretório de reparo
*.\winnt\Program Files\	ED	Programas
*.\temp\	D	Arquivos temporários
HKLM\SYSTEM\CurrentControlSet\Services\EventLog\[Log Name]	ED	Chaves de registro

Legenda: HKLM = registro local; L = ler; E = escrever; D = deletar; X = executar

Após escolher o que deve ser auditado, é necessário implantar as decisões no sistema. Como padrão, o NT disponibiliza poucas ferramentas com esse fim. Para realizar os testes do protótipo do agente sensor de *host*, utilizou-se a configuração existente no "Gerenciador de Usuários para Domínio" - Diretiva de Auditoria (Figura 4.14) e Diretiva de Contas (Figura 4.15) (NT, 1999a).

Figura 4.14 - Diretiva de auditoria

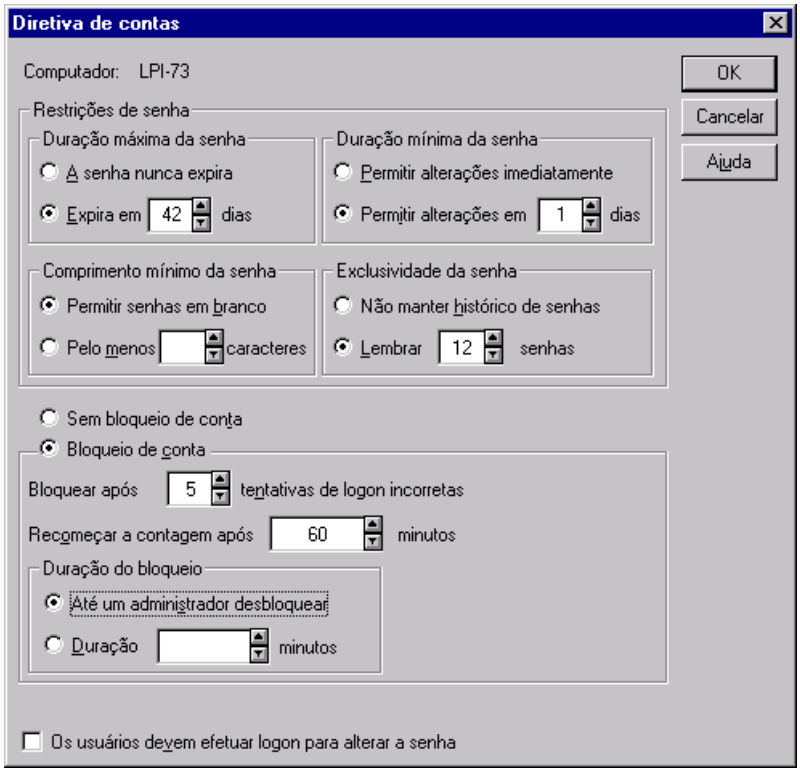


Figura 4.15 - Diretiva de contas

Portanto, para que o sistema de *log* do *Windows* NT guarde um determinado evento de uma aplicação qualquer, é necessário primeiro que a mesma se cadastre no registro do sistema operacional, identificando seu nome e outras informações sob a chave do arquivo de *log* mais adequado. Um exemplo desse cadastro é mostrado na Figura 4.16.

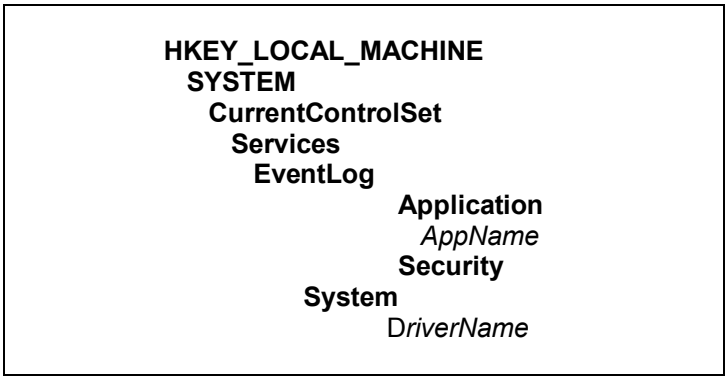


Figura 4.16 – Exemplo de Cadastro

Outra característica do sistema de *log* do *Windows* é a divisão dos eventos em categorias. Por exemplo, o sistema de segurança categorizou os eventos de quatro formas: *Logon/logoff*; acesso ao sistema de arquivo; ações dependentes de privilégio; e mudanças na política de segurança. Para isso, tem-se:

i) identificadores de eventos: números que servem para caracterizar de maneira inequívoca um evento em particular;

ii) arquivos de mensagem: arquivos que contêm descrições sobre os identificadores de eventos, as categorias de eventos e outros parâmetros. Eles são registrados no sistema operacional nas chaves *EventMessageFile*, *CategoryMessageFile* e *ParameterMessageFile* de cada fonte de evento.

Conforme foi comentado anteriormente, a manipulação do sistema de *log* do *Windows* NT é feita através de funções específicas de sua API, descritas na Tabela 4.2.

Tabela 4.2 - API´s de manipulação do sistema de log do Windows NT

API	**Descrição**
OpenEventLog	retorna um identificador para um arquivo de *log* especificado
OpenBackupEventLog	retorna um identificador para um *backup* de arquivo de *log* especificado
RegisterEventSource	retorna um identificador para uma fonte de eventos especificada
DeregisterEventSource	fecha o identificador aberto pela função anterior
CloseEventLog	fecha o identificador aberto por *OpenEventLog*
ReadEventLog	realiza a leitura de determinado número de entradas no arquivo de *log* especificado por um identificador
ReportEvent	escreve um evento no arquivo de *log* especificado por um identificador
BackupEventLog	faz uma cópia de segurança do arquivo de *log* especificado
ClearEventLog	limpa um arquivo de *log* especificado
GetOldestEventLogRecord	retorna o número absoluto do registro mais antigo de um arquivo de *log* especificado
GetNumberOfEventLogRecords	retorna a quantidade de registros de um arquivo de *log* especificado
NotifyChangeEventLogRecord	permite que uma aplicação receba uma notificação quando um evento é escrito em arquivo de *log* especificado ou quando esse é apagado

Quando uma aplicação chama a função *ReportEvent* para escrever uma entrada no arquivo de *log* adequado, o *Windows* NT passa os parâmetros dessa função para o serviço de *log*. Então, esse serviço usa as informações recebidas para preencher uma estrutura EVENTLOGRECORD no arquivo especificado. Essa estrutura corresponde à unidade mínima gravada nos *logs*. A Figura 4.17 ilustra o processo.

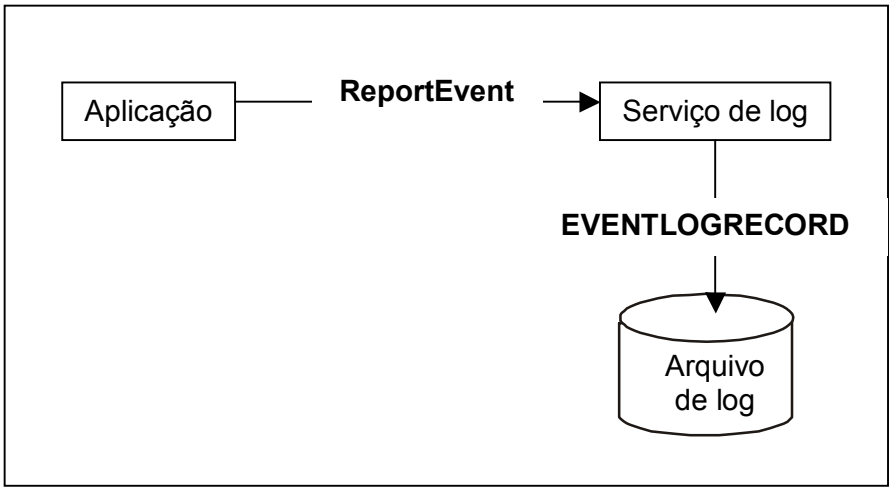


Figura 4.17 - Modelo de escrita do *log* do *Windows* NT

Quando uma aplicação deseja visualizar o conteúdo de um arquivo de *log*, essa deve usar primeiramente a função *OpenEventLog* para conseguir um identificador para o arquivo de *log*. Depois ela deve chamar *ReadEventLog* para ler o evento especificado. Essa ultima função retorna um *buffer* contendo uma estrutura EVENTLOGRECORD e algumas outras informações. O processo é mostrado na Figura 4.18.

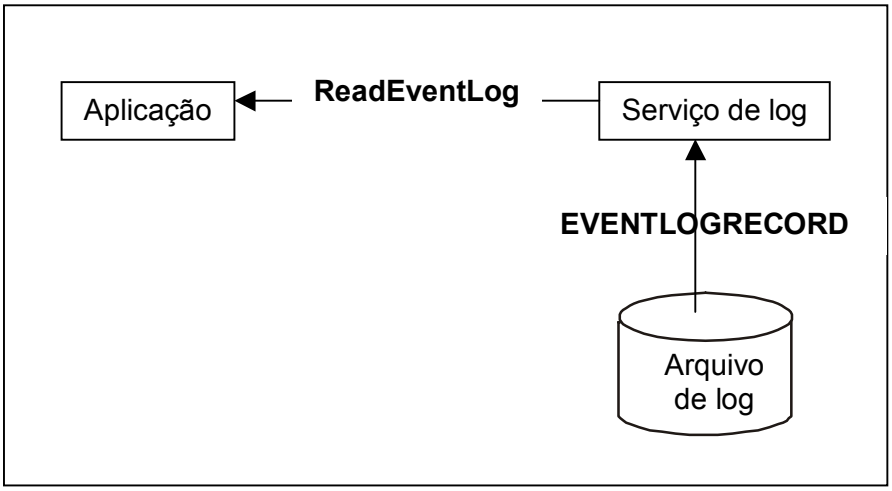


Figura 4.18 - Modelo de leitura do *log* do *Windows* NT

Os estudos desses mecanismos são importantes para a implementação do programa externo do agente sensor de *host*, que é baseado na utilização da API do *Windows*.

4.8.2 Detalhes da implementação

O programa externo do agente sensor de *host* interage diretamente com o sistema de *log* do *Windows* NT. Ele monitora os eventos do sistema em busca daqueles definidos pela política de auditoria como nocivos à segurança. Para realizar a leitura dos registros, ele faz uso de algumas das funções citadas na seção anterior. De posse das informações do registro, o programa então faz uma filtragem dos campos e disponibiliza as informações para que o agente de monitoramento possa analisá-las.

Esse processo pode ser melhor visualizado na Figura 4.19. Ela mostra o diagrama de colaboração das principais classes utilizadas no programa e o sistema de *log*.

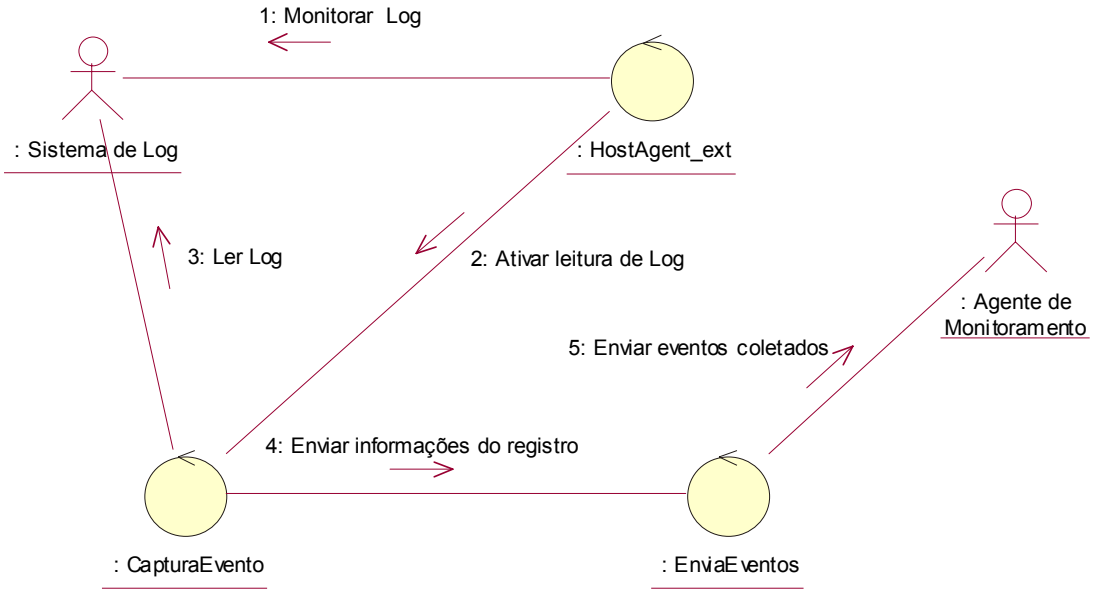


Figura 4.19 - Diagrama de colaboração do programa externo do agente sensor de *host*

Primeiramente, um objeto da classe *HostAgent_ext* é responsável por monitorar o arquivo de *log* de segurança para verificar a inserção de um novo registro. Quando essa é percebida, uma mensagem de ativação de leitura é disparada para o objeto do tipo *CapturaEvento*.

Esse objeto então verifica se o registro auditado é relevante para o escopo do problema, ou seja, se ele está relacionado com alguma questão crítica de segurança. Em caso afirmativo, as informações são resgatados do sistema de *log,* formatadas e enviadas para uma instância da classe *EnviaEventos,* para serem enviadas para o agente SMA.

HostAgent_ext é a classe responsável por monitorar o *log* de segurança do sistema e iniciar o módulo de recuperação do último registro armazenado. Para isso, quatro funções da API do *Windows* são requeridas: *OpenEventLog*, *CreateEvent*, *NotifyChangeEventLog* e *WaitForSingleObject* (Figura 4.20).

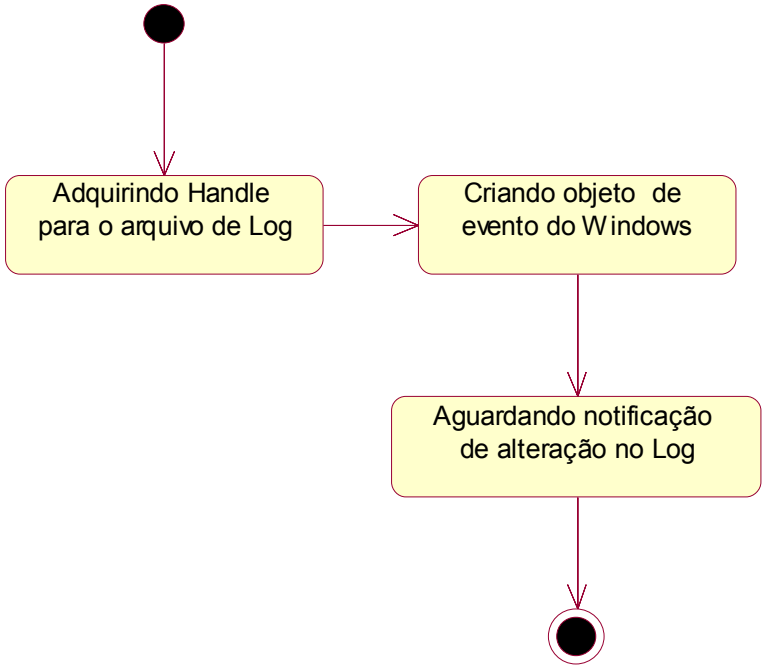


Figura 4.20 - Diagrama de estados de *HostAgent_ext*

A princípio, é preciso adquirir um identificador para o arquivo de *log* de segurança, usando-se a função *OpenEventLog*. É através dele que o programa consegue efetuar operações no referido arquivo. Em seguida, um Objeto de Eventos (OE) do *Windows* é criado através da função *CreateEvent*. Esse objeto funciona como um semáforo: quando ele está verde (sinalizado) as funções que dependem dele podem ser executadas e quando ele está vermelho (não-sinalizado), as funções devem ficar em estado de espera. Ele opera em conjunto com a função *NotifyChangeEventLog*, que é responsável por sinalizá-lo quando uma alteração no *log* é percebida. Assim, o programa fica em estado de espera (através de *WaitForSingleObject*), gastando um mínimo de processamento, até que o OE seja verdadeiro. Quando isso acontece, um objeto da classe *CapturaEvento* é criado (Figura 4.21).

CapturaEvento é a classe diretamente responsável pela recuperação do último registro armazenado no arquivo de *log* de segurança. Através dela, é chamada uma dll (*dynamic linked library*), desenvolvida utilizando-se a linguagem C, especificamente para interagir com o sistema de *log* do Windows e atribuir valores a um objeto da classe EVENTLOGRECORD (usada para armazenar as estruturas EVENTLOGRECORD passadas pelo sistema de *log*). A tarefa da *dll* pode ser resumida da forma que segue.

Inicialmente, é aberto um identificador utilizando-se *OpenEventLog*. Depois, usando-se as funções *GetOldestEventLogRecord* e *GetNumberOfEventLogRecords* (descritas na Tabela 4.2) são obtidos o número absoluto do último registro do *log* e a quantidade de registros existentes. Esses valores são necessários para se calcular a posição da última entrada do *log*, ou seja, aquela que contém as informações do evento que se deseja analisar.

Então, é chamada a função *ReadEventLog*, que recupera as informações do *log* e verifica se o evento é relevante. Em caso positivo, as informações essenciais são selecionadas e formatadas, para serem repassadas para a classe *EnviaPacotes*. Os campos mais importantes são: ano, mês, dia, hora, minuto, segundo, identificador de evento, tipo do evento, categoria do evento, tipo de *login*

efetuado (local ou remoto), nome de quem fez a ação, domínio de quem fez a ação, nome de quem sofreu a ação e o domínio de quem sofreu a ação.

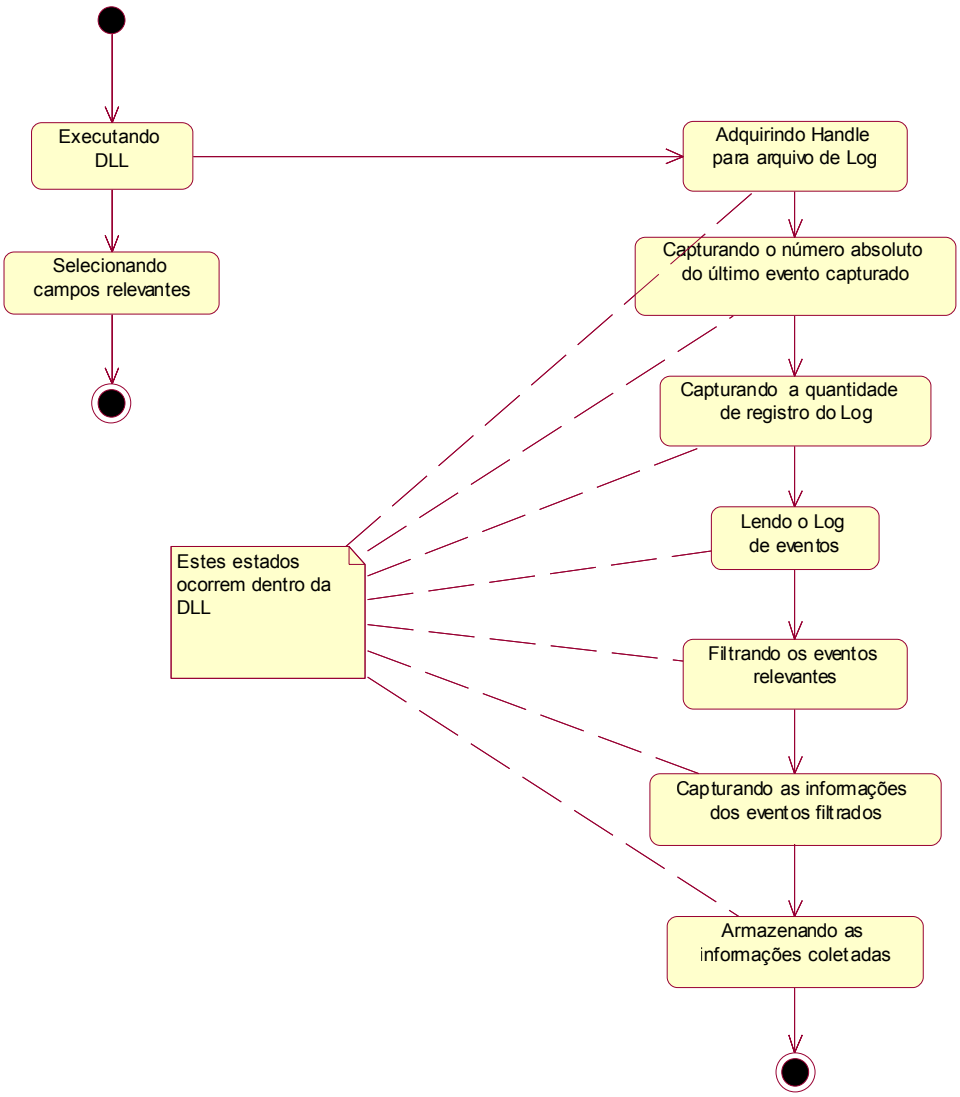


Figura 4.21 - Diagrama de estado de *CapturaEvento*

A classe *EnviaEventos* possui o mesmo princípio de funcionamento da classe *EnviaPacotes* do agente sensor de rede (Figura 4.13). Quando uma quantidade arbitrária de registros está contida no vetor, as informações dos eventos são enviadas para os agentes especializados. Ela é escrita inteiramente em Java, não requerendo nenhuma função da API do sistema operacional.

5 RESULTADOS PARCIAIS DE SIMULAÇÕES COM OS SENSORES DE REDE E HOST

Neste capítulo são apresentados os resultados das simulações realizadas com os agentes sensores de rede e de *host*. Para isso, é apresentado o protótipo implementado desses agentes, destacando-se sua interação com o agente de monitoramento e demonstrando os resultados obtidos nas simulações.

5.1 Resultado da comunicação inter-agentes

Os testes com o protótipo implementado foi testado nos laboratórios da UFMA, onde encontra-se atualmente instalado e em execução. Para compor o cenário da comunicação, o agente de monitoramento (SMA) foi dotado da habilidade de ativar os agentes sensores e disponibilizar m janelas as informações enviadas por eles (Figura 5.1). Com essa modificação, entende-se que o agente de monitoramento fará uma das tarefas do agente de segurança local, apresentado na Seção 3.3.

Quando os agentes sensores e SMA são criados, eles são registrados no Agente Servidor de Nomes. Paralelo a isso, esses agentes também registram suas habilidades na Base de Conhecimento do Agente Facilitador, para poderem ser utilizadas pelos demais agentes da sociedade. O Facilitador questiona as habilidades dos agentes em intervalos de tempo regulares, de acordo com a configuração adotada na Seção 4.4 (Figura 5.2).

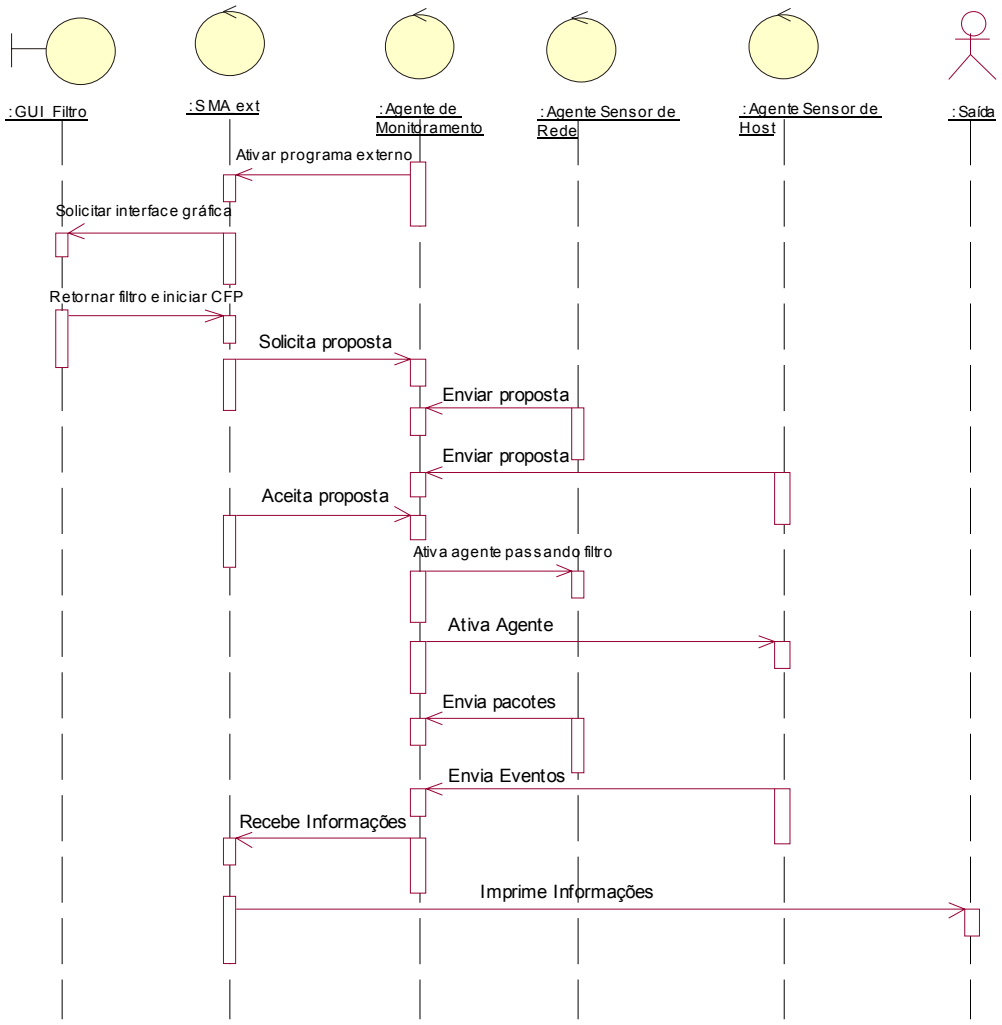


Figura 5.1 - Diagrama de seqüência do agente de monitoramento (notação UML)

Ao ser inicializado, o agente de monitoramento ativa o seu programa externo (classe *SMA_ext*), que, por sua vez, abre uma interface gráfica, para que o administrador de segurança possa fossa fornecer parâmetros de filtragem para serem utilizados pelo agente sensor de rede (Figura 5.3). Nessa mesma interface o administrador dá início a todo o processo que será descrito a seguir.

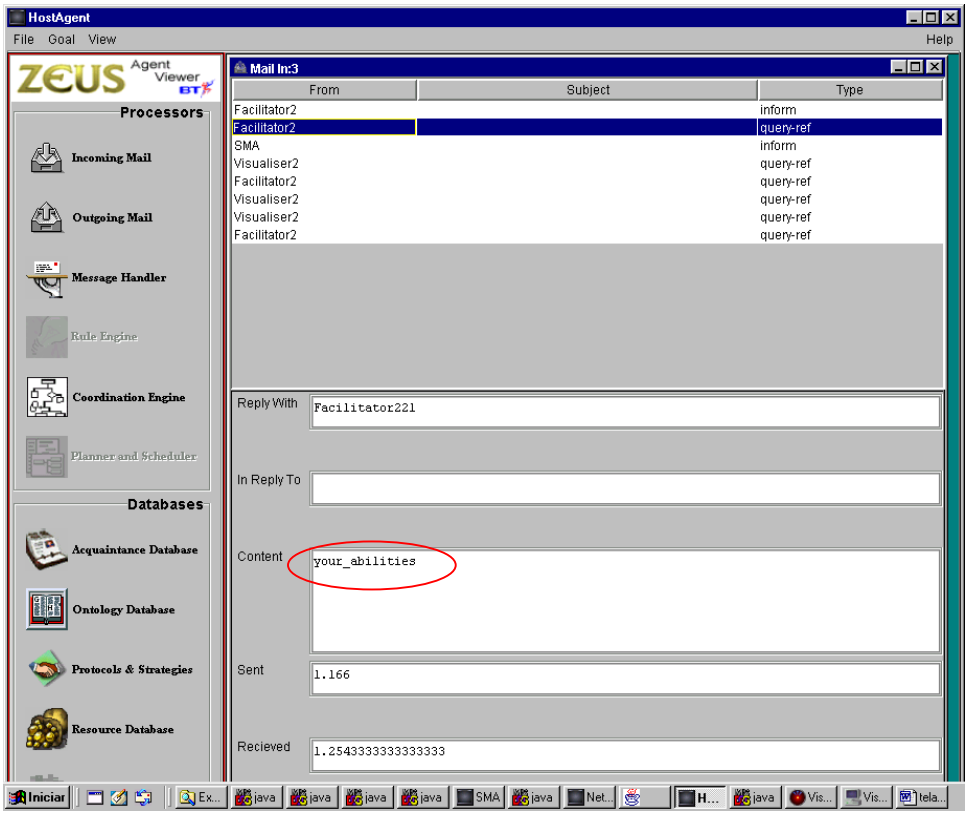


Figura 5.2 - Solicitando as habilidades dos agentes

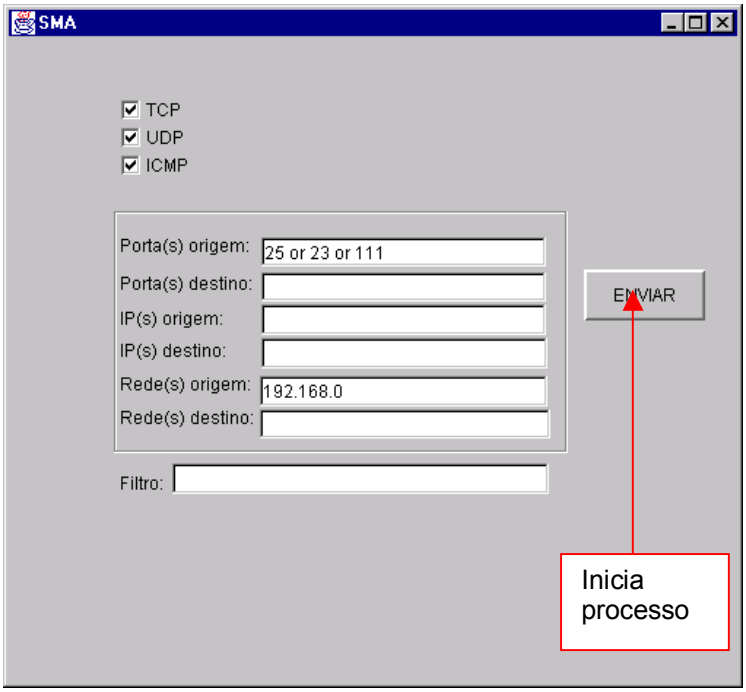


Figura 5.3 - Interface gráfica do agente de monitoramento

Para o agente SMA realizar o seu objetivo, que é produzir **Saída**, conforme foi especificado anteriormente, esse busca no Agente Facilitador os recursos exigidos para tal tarefa (**Pacotes** e **Eventos**), ou seja, questiona qual agente possui habilidade de capturar pacotes e qual possui a habilidade de capturar eventos. Esse, por sua vez, retorna o nome do agente associado à sua habilidade, conforme mostra a Figura 5.4.

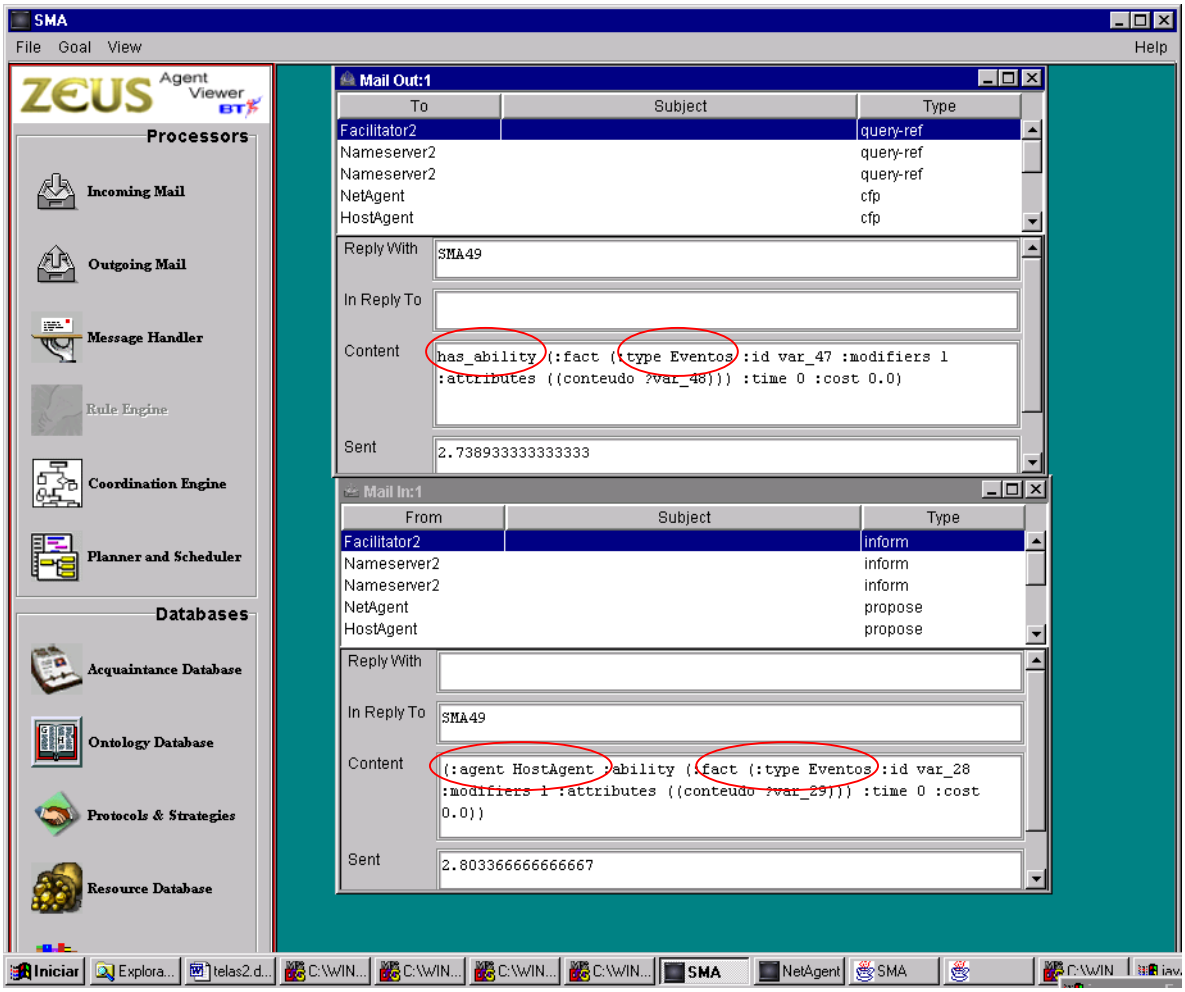


Figura 5.4 - Encontrando os agentes com habilidades exigidas

Conhecendo o nome dos agentes com as habilidades exigidas, o agente de monitoramento solicita seus endereços ao ANS (Figura 5.5), que responde com uma mensagem do tipo *inform* da classe *Performative* (Apêndice A) , contendo o nome e o endereço IP da máquina onde os agentes solicitados estão instalados (Figura 5.6).

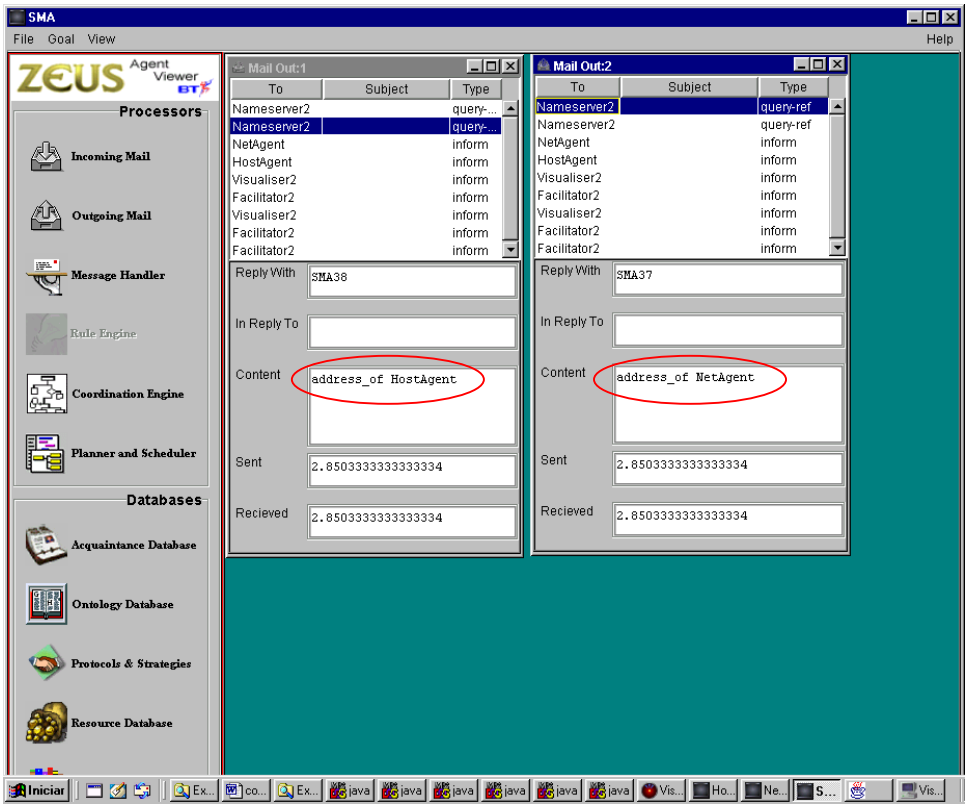


Figura 5.5 - Solicitando endereços ao Agente Servidor de Nomes

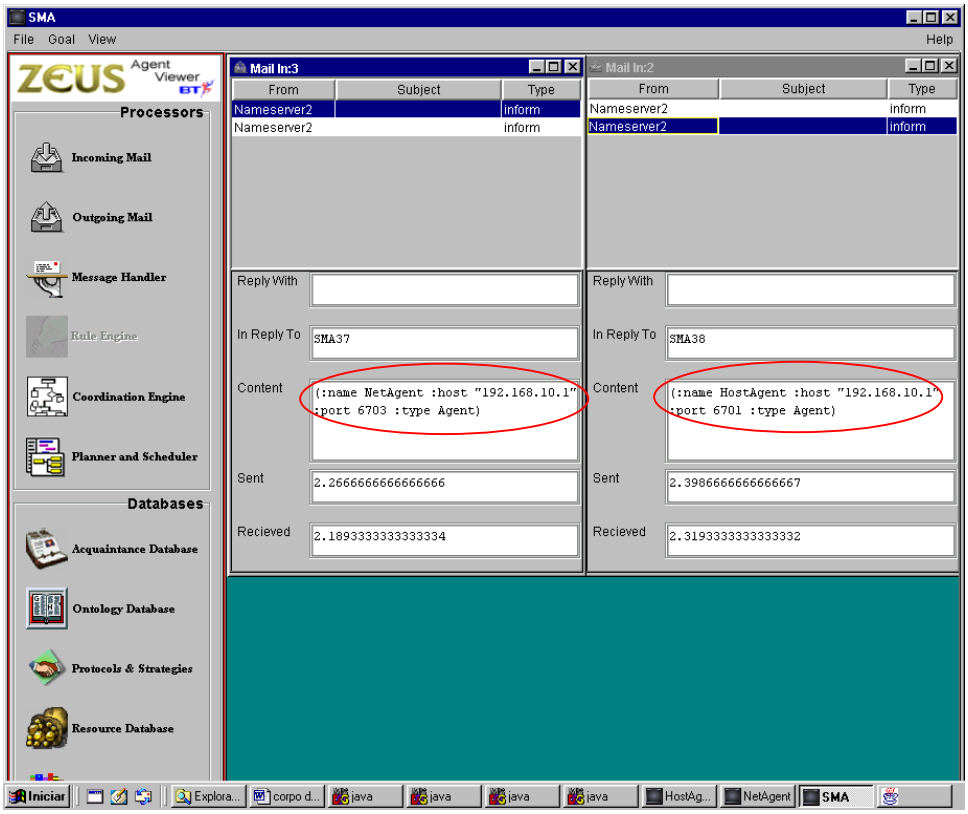


Figura 5.6: Enviando os endereços dos agentes

Para iniciar o mecanismo de contratação (Apêndice A, Figura A5) o agente SMA anuncia um contrato para a sociedade através de chamada para proposta (mensagem *cfp*), e os agentes NetAgent e *Host*Agent lançam suas propostas para serem analisadas (Figura 5.7). Caso tais propostas sejam aceitas, é então enviada uma mensagem de proposta aceita (*accept-proposal*) aos agentes sensores.

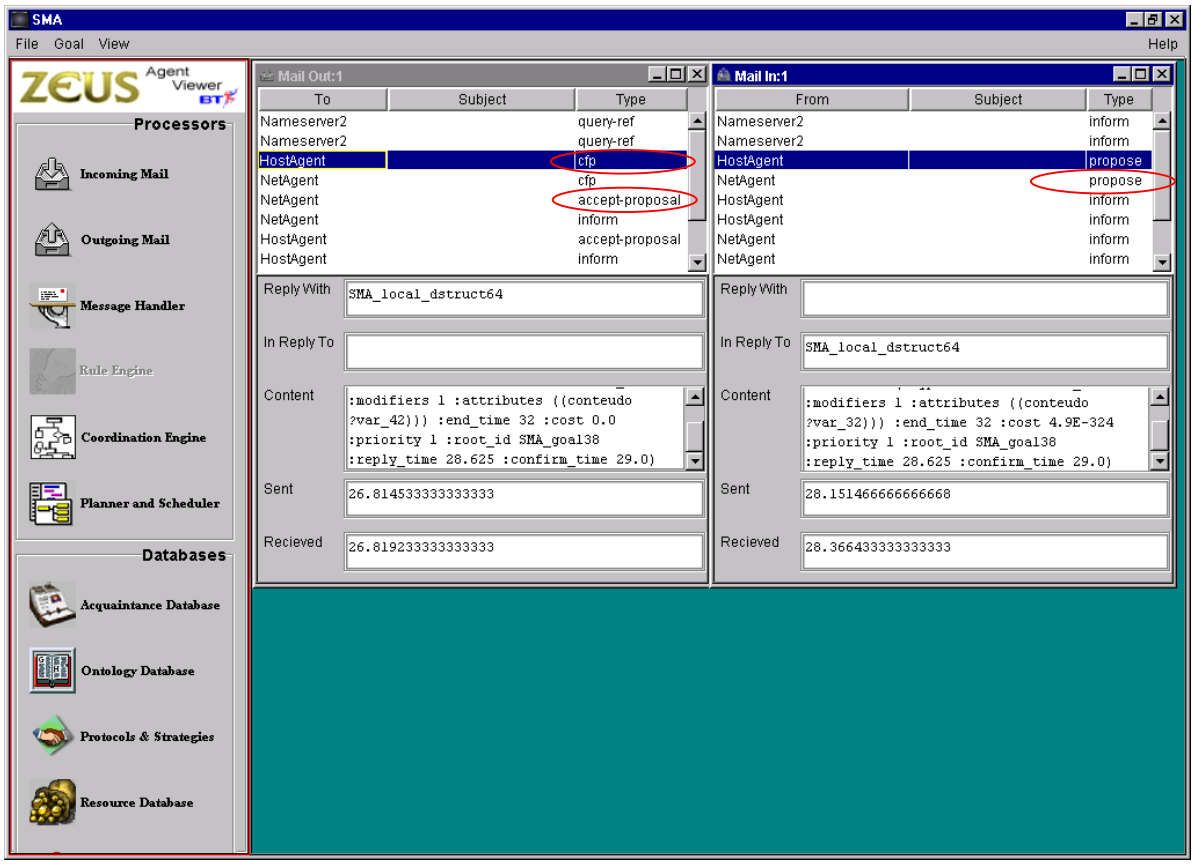


Figura 5.7 - Mensagem *cfp*

Seguindo a seqüência de mensagem, tem-se que após receberem a confirmação de proposta aceita, os agentes sensores NetAgent e *Host*Agent finalizam a negociação e iniciam suas tarefas para se atingir a meta principal do agente de monitoramento (produzir **Saída)**, baseada no recebimento dos pacotes que estão trafegando na rede e dos eventos que estão ocorrendo no *host* e, em seguida, mostrá-los na tela (Figura 5.8). Ressalta-se ainda que ao ser ativado, o agente sensor de rede recebe seus parâmetros de filtragem, que foram configurados na interface gráfica da Figura 5.3.

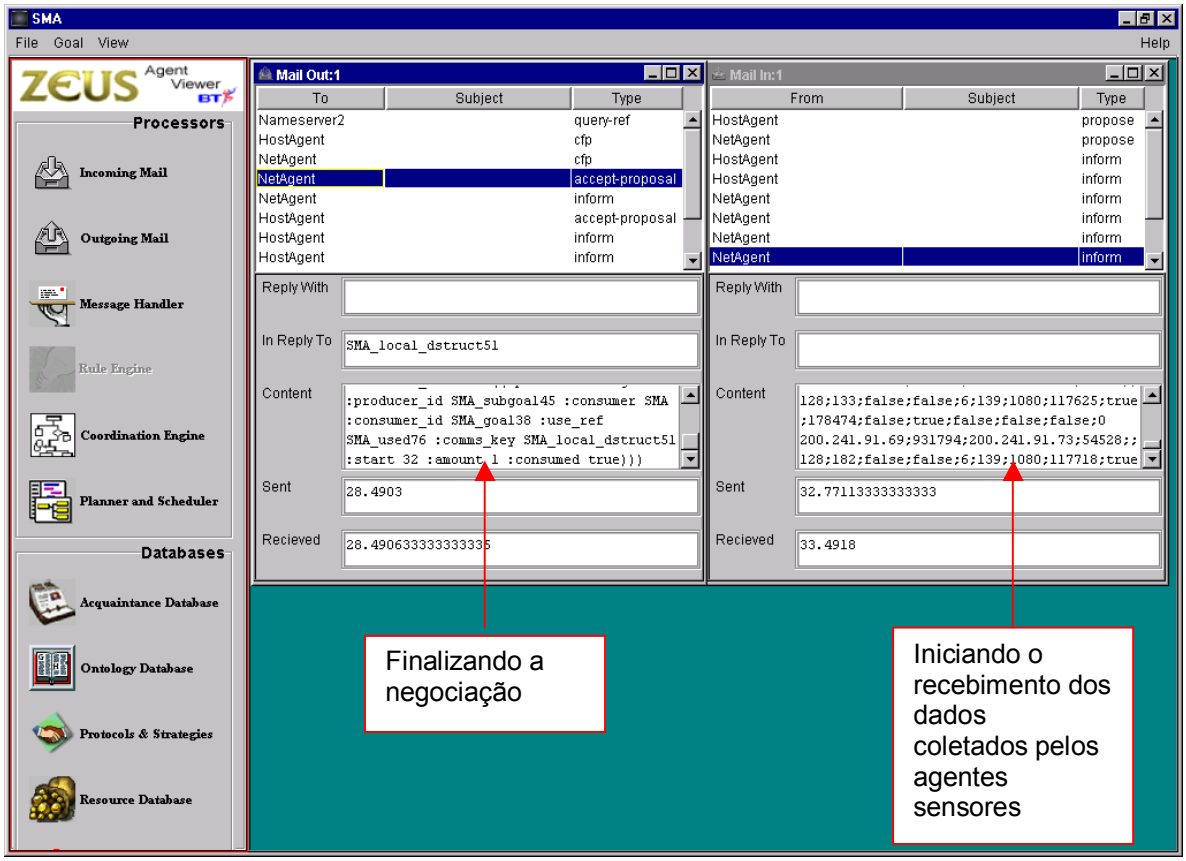


Figura 5.8 - Finalizando a negociação e iniciando a tarefa do agente SMA

Em execução, os agentes sensores de rede e de *host* enviam, simultaneamente, os seus eventos coletados ao agente de monitoramento. (Figuras 5.9 e 5.10).

Observa-se que o conteúdo da mensagem recebida pelo agente de monitoramento contém as informações coletadas pelos agentes sensores. Essas informações serão detalhadas nas seções seguintes.

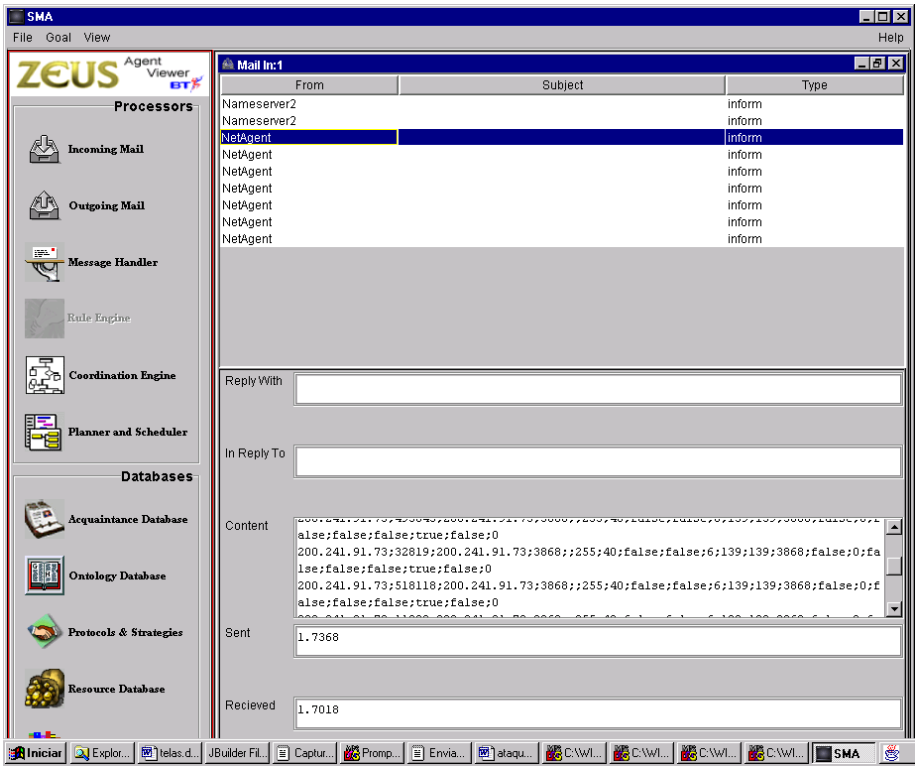


Figura 5.9 – Enviando pacotes ao SMA

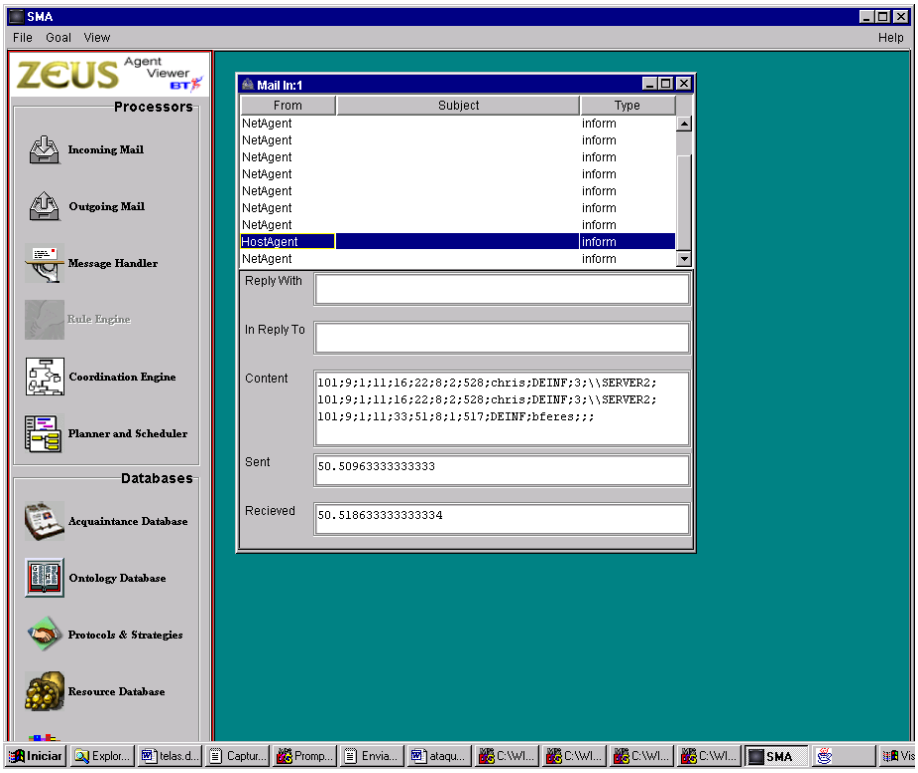


Figura 5.10 - Enviando eventos ao SMA

5.2 Dados gerados pelo agente sensor de rede

Para testar a eficiência do agente sensor de rede em relação à sua habilidade em capturar informações da rede, mediante a utilização de filtros, foram utilizadas algumas ferramentas disponíveis na internet para simular tentativas de ataques a estações de trabalho dentro de um ambiente real de produção, em que utilizam diversos protocolos, tais como TCP/IP e IPX/SPX[28]. Os endereços IP de origem e destino das máquinas envolvidas nos testes foram alterados, para serem preservados.

Os campos do pacote capturado podem mudar de acordo com o protocolo envolvido na comunicação, baseados na arquitetura TCP/IP. Como todos os pacotes capturados são IP, então os primeiros campos são padronizados conforme mostrado na Tabela 5.1.

Tabela 5.1 - Campos do pacote IP capturado pelo agente sensor de rede

1º campo	2º campo	3º campo	4º campo	5º campo	6º campo	7º campo	8º campo	9º campo
Endereço de origem	Hora do pacote (μs)	Endereço de destino	Identificação do pacote	Tempo de vida do pacote	Tamanho do pacote IP	Pacote não fragmentado	Mais fragmentos no pacote	Identificação do Protocolo

Dependendo da identificação do protocolo – 6 (TCP), 17 (UDP) ou 1 (ICMP) - os demais campos são diferentes (Tabela 5.2).

Tabela 5.2 - Campos para os protocolos TCP, UDP e ICMP do pacote capturado

Campo / Protocolo	10º campo	11º campo	12º campo	13º campo	14º campo	15º campo	16º campo	17º campo	18º campo	19º campo	20º campo
TCP (6)	Porta de origem	Porta de destino	Seqüência do pacote	Flag ACK	Número ACK	Flag FIN	Flag PSH	Flag RST	Flag SYN	Flag URG	Ponteiro urgente
UDP (17)	Porta de origem	Porta de destino	Tamanho do Pacote UDP								
ICMP (1)	Tipo	Código									

[28] Protocolo utilizado no sistema operacional Novell/Netware.

No primeiro teste realizado simulou-se um ataque do tipo *Land* (Figura 5.11). Com esse exemplo, pode-se verificar o uso de técnicas associadas (IPSPOOFING e SYNFLOOD), conforme explanado na Seção 4.7.1.

```
192.168.10.30;224156;192.168.10.30;3868;255;40;false;false;6;139;139;3868;false;0;false;false;false;true;false;0
```

Figura 5.11 - Ataque *Land*

O ataque Land foi identificado através dos endereços de origem e destino iguais (campos 1 e 3, respectivamente), protocolo TCP (campo 9), porta 139 de origem e destino da conexão (campos 10 e 11) e flag SYN ativado (campo 18).

O teste seguinte foi feito, simulando-se o ataque *Winnuke*, que pode causar negação de serviço em uma máquina Windows 95 (Figura 5.12). Esse ataque consiste em enviar uma string OOB (Out Of Band) no ponteiro urgente (campo 19 – flag URG ativado e campo 20 diferente de zero) em uma conexão TCP estabelecida, avisando ao destinatário que os dados contidos nesse segmento devem ser processados com urgência, independente de sua posição na fila.

Como alguns sistemas operacionais não tratam essa informação, diversos problemas podem ocorrer, tais como erro de GPF (tela azul), além de poderem apresentar falhas em conexões de rede até que a máquina seja reiniciada.

```
192.168.10.30;729354;192.168.10.33;56259;64;60;false;false;6;1057;139;2609292554;false;0;false;false;false;true;false;0
192.168.10.30;729757;192.168.10.33;56260;64;40;false;false;6;1057;139;2609292555;true;338618;false;false;false;false;false;0
192.168.10.30;730026;192.168.10.33;56261;64;43;false;false;6;1057;139;2609292555;true;338618;false;true;false;false;true;3
192.168.10.30;730368;192.168.10.33;56262;64;40;false;false;6;1057;139;2609292558;true;338624;false;false;false;false;false;0
192.168.10.30;834653;192.168.10.33;56263;64;40;false;false;6;1057;139;2609292558;true;338624;false;false;true;false;false;0
```

Figura 5.12: Ataque *Winnuke*

Verificou-se, também, a varredura de *ping* e a varredura de portas. Essas técnicas são muito utilizadas por usuários maliciosos e administradores de segurança para coletar informações de uma rede, quanto ao nome de máquinas e seus respectivos IP, quais portas (*telnet*, *ftp*, *smtp*, *finger*, etc) estão abertas etc. Os resultados são mostrados nas Figuras 5.13 e 5.14.

```
192.168.10.30;140349;192.168.10.1;43573;128;64;false;false;1;8;0
192.168.10.30;141736;192.168.10.2;43829;128;64;false;false;1;8;0
192.168.10.30;144196;192.168.10.3;44085;128;64;false;false;1;8;0
192.168.10.30;155040;192.168.10.5;45109;128;64;false;false;1;8;0
192.168.10.30;155945;192.168.10.6;45365;128;64;false;false;1;8;0
```

Figura 5.13 - Varredura de Ping

Um dos passos básicos para o mapeamento de uma rede é realizar uma varredura de ping automatizada. O *ping* é tradicionalmente usado para enviar pacotes ICMP ECHO (protocolo 1, tipo 8 e código 0) para um sistema alvo, em uma tentativa de obter um ICMP ECHO_REPLY (protocolo 1, tipo 0 e código 0), que indica que o sistema-alvo está ativo.

Entretanto, quando o tráfego ICMP é bloqueado[29], a varredura de porta (*port scanning*) é a técnica a ser usada para determinar *hosts* ativos. Existem sistemas que realizam varredura de *ping* TCP, que consiste em enviar pacotes TCP SYN, tendo como resposta pacotes TCP SYN/ACK no caso dos *host*s estarem ativos. Esse sistema usa, como padrão, a porta 80, que tem acesso permitido na maioria dos roteadores e firewalls, por se tratar da porta de serviço dos servidores WEB (http).

[29] Muitas empresas preocupadas com segurança, bloqueiam o ICMP no firewall.

192.168.10.30;351845;192.168.10.73;32065;128;44;false;false;6;1817;140;359797;false;0;false;false;false;true;false;0

192.168.10.30;822581;192.168.10.73;32321;128;44;false;false;6;1817;140;359797;false;0;false;false;false;true;false;0

192.168.10.30;323374;192.168.10.73;32577;128;44;false;false;6;1817;140;359797;false;0;false;false;false;true;false;0

192.168.10.30;824134;192.168.10.73;32833;128;44;false;false;6;1817;140;359797;false;0;false;false;false;true;false;0

192.168.10.30;829232;192.168.10.73;33089;128;44;false;false;6;1818;139;359801;false;0;false;false;false;true;false;0

192.168.10.30;829578;192.168.10.73;33345;128;40;false;false;6;1818;139;359802;true;1111647;false;false;false;false;false;0

192.168.10.30;845272;192.168.10.73;33601;128;40;false;false;6;1818;139;359802;true;1111647;true;false;false;false;false;0

192.168.10.30;850999;192.168.10.73;34113;128;44;false;false;6;1819;138;359811;false;0;false;false;false;true;false;0

192.168.10.30;324943;192.168.10.73;34369;128;44;false;false;6;1819;138;359811;false;0;false;false;false;true;false;0

192.168.10.30;825714;192.168.10.73;34625;128;44;false;false;6;1819;138;359811;false;0;false;false;false;true;false;0

192.168.10.30;326502;192.168.10.73;34881;128;44;false;false;6;1819;138;359811;false;0;false;false;false;true;false;0

192.168.10.30;332584;192.168.10.73;35137;128;44;false;false;6;1820;137;487811;false;0;false;false;false;true;false;0

.....

Figura 5.14 - Varredura de Porta

A varredura de portas é o processo de se conectar a portas TCP e UDP do sistema-alvo para determinar quais serviços estão em execução ou em estado de escuta (*listening*). A identificação de portas “escutando” é crucial para determinar o tipo de SO e aplicativos que são utilizados na rede-alvo. Existem vários tipos de varredura de portas, a saber: a varredura de conexão TCP, a varredura TCP SYN, a varredura TCP FIN, a varredura TCP de árvore de natal, varredura TCP nula e a varredura UDP (MCCLURE et al, 2000).

A técnica utilizada na simulação foi a de varredura de conexão TCP, que consiste em se conectar à porta alvo e tentar completar um *handshake* de 3 etapas (SYN, SYN/ACK e ACK). Analisando a Figura 5.14, conclui-se que o *host* 192.168.10.73 está com a porta 139 aberta, a qual garante uma vulnerabilidade que pode ser explorada por invasores.

5.3 Dados gerados pelo agente sensor de *host*

A fim de demonstrar como o agente sensor de *host* está interagindo com o sistema de log do Windows, foram feitas várias simulações, verificando-se o comportamento, bem como os resultados obtidos dos testes. Os nomes de usuários, máquinas e domínios foram modificados, para serem preservados.

O padrão de saída do agente sensor de *host* é composto de 13 campos, no máximo. A Tabela 5.3 mostra os campos comuns a todos os eventos capturados. Os demais campos (10º ao 13º) variam de acordo com o identificador do evento e serão demonstrados posteriormente.

Tabela 5.3 – Campos da string de eventos

1º campo	2º campo	3º campo	4º campo	5º campo	6º campo	7º campo	8º campo	9º campo
Ano	Mês	dia	Hora	Minuto	Segundo	Tipo de Evento[30]	Categoria do Evento[31]	Identificação do Evento

Para fins de comprovação da funcionalidade do agente, foram realizados testes de: sucesso e falha de logon; limpeza de log; criação e deleção de contas de usuários; mudança de auditoria e diretiva de sistema. Os resultados serão apresentados a seguir:

***i)* logon com sucesso**

Um evento de logon com sucesso, tem como saída a Figura 5.15.

```
101;9;1;11;16;22;8;2;528;joao;DOMAIN_TESTE;3;\\SERVER_TESTE
```

Figura 5.15 - Evento de logon remoto com sucesso

Verificando-se a seqüência dos campos, têm-se: um evento de sucesso de auditoria (código 8), da categoria Logon/Logoff (código 2), identificado através do código 528 como logon com sucesso do usuário “joão”, pertencente ao domínio

[30] Se sucesso ou falha de auditoria, vistos na Seção 4.8.1.
[31] *Login/logoff*, acesso a arquivos e objetos e os demais apresentados na Seção 4.8.1

“DOMAIN_TESTE”, sendo um logon do tipo remoto (código 3) e, por último, tem-se a informação do nome do servidor do domínio (\\SERVER_TESTE), ao qual o logon foi efetuado.

Em caso de logon local (código 2), o último campo é preenchido com o nome do servidor, sem o “\\”, conforme demonstrado na Figura 5.16. Dessa forma, pode-se verificar se um usuário obteve acesso físico (local) ou remoto a um servidor crítico, em desacordo com a política de segurança da instituição.

```
101;9;1;11;16;22;8;2;528;joao;DOMAIN_TESTE;2;SERVER_TESTE
```

Figura 5.16 - Evento de logon local com sucesso

Contudo, muitos sistemas de detecção de intrusos não monitoram eventos de sucesso de logon, pois é gerada uma sobrecarga demasiada no arquivo de log do SO, à proporção que o número de usuários cadastrados cresce e, conseqüentemente, haverá uma sobrecarga no próprio SDI, que terá que analisar todas as ocorrências desse evento. Além disso, restrições de horários, acesso físico, correção de vulnerabilidades, entre outros, pode não ser função do SDI e sim do administrador de segurança. Para tanto, os SDI´s analisam um conjunto de eventos e não um evento isolado para conseguir detectar uma tentativa de intrusão.

ii) falha de logon

Seguindo a mesma analogia têm-se, como exemplo, dois tipos de eventos tratados pelo agente sensor de *host*: o de falha de logon por motivo de usuário ou senha incorreta e falha de logon devido a conta bloqueada. Os resultados são relatados na Figura 5.17.

```
101;9;1;12;46;36;16;2;529;joao;DOMAIN_TESTE;2;SERVER_TESTE
101;9;1;12;47;1;16;2;539;joao;DOMAIN_TESTE;2;SERVER_TESTE
```

Figura 5.17 - Evento de falha de logon

Os dois são eventos de falha de auditoria (código 16), da categoria Logon/Logoff, identificados como falha de logon por motivo de usuário ou senha

incorreta, através do código 529 e falha de logon devido à conta bloqueada, código 539, do usuário “joão”, pertencentes ao domínio “DOMAIN_TESTE”, sendo o logon do tipo local (código 2) e, por fim, tem-se a informação do nome do servidor do domínio (SERVER_TESTE), ao qual a tentativa de logon foi efetuada.

Em um registro de segurança cheio de eventos 529 ou 539 há evidências de um ataque de adivinhação de senhas automatizado, como por exemplo, um ataque NAT (*NetBIOS Auditing Tool*), que consiste em tentar adivinhar senhas de um sistema alvo a partir de um “array” predefinido e de listas fornecidas pelo usuário.

***iii)* limpeza de log**

Outro evento importante a ser analisado é o de limpeza de log, pois invasores um pouco mais experientes constumam apagar as trilhas de auditorias, dificultando o trabalho dos administradores de segurança (Figura 5.18).

```
101;9;1;11;33;51;8;1;517;joao;DOMAIN_TESTE;;
```

Figura 5.18 - Evento de limpeza de log

Analisando o evento, conclui-se que ocorreu um sucesso de auditoria, da categoria “acesso a arquivos e objetos” (código 1), identificado por limpeza de log (código 517), realizado no domínio DOMAIN_TESTE. Os campos subseqüentes não são preenchidos, pois se parte do princípio de que o usuário deve estar conectado ao sistema e ter privilégios para realizar a tarefa e, portanto, vários eventos são disparados antes, tais como o de logon com sucesso, que fornece as informações omitidas no evento detectado.

***iv)* criação e deleção de conta de usuário**

Uma vez que um atacante tenha obtido acesso de administrador em um sistema, eles tentam impedir toda e qualquer detecção adicional de sua presença. Para isso, eles podem deletar, criar ou alterar contas de usuários, dentre outras formas de garantir acesso fácil no futuro.

Os eventos de criação e deleção de conta de usuário são monitorados pelo agente sensor de *host*, conforme a Figura 5.19.

Criação de conta

101;9;1;12;18;53;8;7;624;joao;DOMAIN_TESTE;; DOMAIN_TESTE;teste1
101;9;1;12;20;32;8;7;624;joao;DOMAIN_TESTE;; DOMAIN_TESTE;teste2
101;9;1;12;20;39;8;7;624;joao;DOMAIN_TESTE;; DOMAIN_TESTE;teste3

Deleção de conta

101;9;1;12;23;42;8;7;630;joao;DOMAIN_TESTE;; DOMAIN _TESTE;teste1
101;9;1;12;23;47;8;7;630;joao;DOMAIN_TESTE;; DOMAIN _TESTE;teste2
101;9;1;12;23;52;8;7;630;joao;DOMAIN_TESTE;; DOMAIN _TESTE;teste3

Figura 5.19 - Evento de criação e deleção de conta de usuário

Os eventos são da categoria “gerenciamento de usuários e grupos (código 7), identificados como criação (624) e deleção (630) de contas de usuários. Sendo esses realizados pelo usuário “joao” do domínio DOMAIN_TESTE, adicionados e deletados do domínio DOMAIN_TESTE, sendo que as contas que foram criadas e deletadas foram: teste1, teste2 e teste 3.

Ressalta-se, ainda, que outros eventos são disparados quando da criação e deleção de usuários, tais como: adição ou remoção do usuário a grupos locais e globais (NT, 1999).

***v)* mudança de auditoria e diretiva**

Um dos artifícios bastante utilizados pelos atacantes é tentar apagar seus rastros no sistema, adulterando ou desligando os registros de auditoria. Portanto, esses eventos foram considerados importantes nesta pesquisa.

Mudança de Auditoria

101;9;1;12;52;41;8;6;612;joao;DOMAIN_TESTE;;

Mudança de Diretiva

101;9;1;13;51;58;8;7;643;joao;DOMAIN_TESTE;;

Figura 5.20 - Evento de mudança de auditoria e diretiva

Os eventos mostrados na Figura 5.20 são da categoria “alterações na auditoria de segurança“ (código 6), por se tratar de uma modificação na política de auditoria do sistema e merece ser verificada e da categoria “gerenciamento de usuários e grupos (código 7), por se tratar de uma mudança de diretiva na auditoria de contas dos usuários, onde são definidas políticas de segurança, tais como o bloqueio de contas de usuários após sucessivas tentativas de logon. Os demais campos registrados são o nome do usuário e o domínio de quem realizou o processo.

5.4 Considerações finais

Os resultados das simulações demonstram os tipos de dados coletados pelos agentes sensores de rede e *host*. Foi feita uma análise significativa das informações geradas por cada agente e apresentando, de maneira geral, como os IDS´s trabalham no reconhecimento de assinaturas de ataques.

Essas informações serão o ponto de partida para a análise e, conseqüentemente, tomada de decisão pelo sistema NIDIA no caso de uma tentativa de ataque ocorrer.

6 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Neste capítulo, apresentam-se as conclusões, através dos principais pontos alcançados com esta dissertação e suas contribuições. Também, delineam-se sugestões de futuras pesquisas em segurança e detecção de intrusões em redes de computadores, utilizando-se sistemas multiagentes.

6.1 Contribuições do trabalho

A segurança da informação constitui problemática relevante na atualidade, ensejando inúmeras tentativas de implementação de mecanismos de proteção mais eficientes, visto que os utilizados não têm conseguido proporcionar o grau de segurança almejado, frustrando a expectativa daqueles que confiam nos seus serviços. Um dos meios mais empregados é o *firewall* que, apesar de possuir muitas vantagens, apresenta algumas limitações que podem possibilitar ataques bem sucedidos a um ambiente.

Nessa perspectiva, uma ferramenta que pode ajudar a elevar o patamar de segurança das instituições e empresas são os Sistemas de Detecção de Intrusão (SDI´s), sendo, atualmente, uma das medidas de segurança mais utilizadas, com um bom nível de satisfação por parte dos usuários.

Baseado nas pesquisas existentes na área de SDI´s, foi proposta, nesta dissertação, como contribuição, uma arquitetura de sistema de detecção de intrusos, baseada na cooperação de agentes inteligentes, associadas ao uso de redes neurais artificiais, para detectar e tratar as tentativas de invasão em uma rede de computadores, englobando as diversas características desejáveis em sistemas dessa natureza.

Por dispor de várias bases de dados, o modelo proposto pode ser facilmente incrementado para novas situações, bem como se adaptar a novas

estratégias, ao reconhecimento de novas assinaturas de intrusão, permitir que o administrador tenha registro de origens duvidosas e incrementar novas ações.

Foi implementado o agente sensor de rede com a capacidade de delimitar o tipo de dado a ser coletado, através da utilização de filtros. Esta é uma contribuição importante, pois dá ao sistema uma grande flexibilidade, devido à facilidade de se contruir agentes sensores de rede com funções diferentes. Por exemplo, um agente poderá ser ativado para capturar pacotes de uma ou mais origem específica, enquanto outro poderá ser ativado para capturar pacotes que correspondam a uma tentativa de intrusão que podem ser capturadas através do processo de filtragem, discutidos na Seção 4.7.

Como contribuição, também foi implementado o agente sensor de host para trabalhar em ambiente Windows, porém a sua estrutura permite que outros agentes sensores de host possam ser implementados para interagirem com sistemas operacionais Unix.

Por fim, obteve-se um protótipo funcional do sistema NIDIA, com a capacidade de coletar dados em tempo real ou em batch e gerar informações necessárias para uma análise consistente, com o propósito de detectar e tratar tentativas de intrusão em uma rede de computadores e em servidores críticos.

6.2 Considerações finais

Com a utilização da plataforma ZEUS, os esforços foram concentrados no problema do domínio, mais precisamente, nas tarefas dos agentes e isso facilitou bastante na obtenção dos resultados. Porém, foram verificadas algumas limitações quanto ao consumo de memória e velocidade de transmissão das mensagens (atualmente 24 mensagens/seg), pois o sistema necessita de mais de 64 MB[32] de RAM para oferecer um bom desempenho.

[32] Foram feitos testes em microcomputadores com até 128 MB e, neste caso, os resultados foram satisfatórios.

Pode-se ampliar o número de mensagens enviadas por segundo, recompilando-se o código ZEUS, mas a garantia de entrega das mensagens pode ficar comprometida. Portanto, para resolver esse problema, que é mais crítico para o agente sensor de rede, optou-se pela captura de pacotes em *batch*, para depois serem enviados ao agente de monitoramento.

Não foi abordada suficientemente, nesta dissertação, a questão relacionada ao desempenho do sistema NIDIA, pois sabe-se que Java é uma linguagem que congrega muitas vantagens no desenvolvimento de sistemas multiagentes, entretanto apresenta problemas de desempenho (Doederlein, 1999) que precisa de uma atenção especial. Dessa forma, será objeto de estudo detalhado em trabalhos futuros.

Quanto à situação atual do sistema NIDIA, os agentes de monitoramento, avaliação e controle de ações já são temas de outras dissertações de mestrado e portanto, estão sendo estudados e implementados. Com isso, os padrões de ataques e as técnicas utilizadas pelos *hackers* e sua interação com os mecanismos de proteção mais comuns, como os *firewalls,* já são frutos de estudos (DIAS, 2001; SANTOS, 2002; SOUSA, 2002; OLIVEIRA, 2002).

6.3 Trabalhos Futuros

Como proposta para trabalhos futuros e sugestões para o enriquecimento deste, pode-se listar:

- construir agentes sensores de rede para ler o conteúdo dos pacotes e não somente os cabeçalhos;
- implementar agentes sensores de *host* para lerem os logs de sistema e aplicação do Windows NT. Dessa forma, pode-se detectar usos indevidos de recursos (memória, UCP, etc), bem como acessos indevidos a aplicações;
- implementar agentes sensores de *hosts* para trabalharem em outras plataformas;

- no Windows NT, transformar os agentes sensores de rede e de *host* em serviços, de forma que toda vez que o sistema for reiniciado, os agentes estarão prontos para serem ativados;
- construir agentes para fazer a detecção por anomalia, tornando o sistema NIDIA um sistema híbrido;
- usar criptografia na comunicação inter-agentes;
- aperfeiçoar a aquisição de dados, com o uso de novas tecnologias, visando melhorar o desempenho do sistema;
- implementar o mesmo protótipo do modelo desenvolvido neste trabalho porém, utilizando outras plataforma de construção de agentes, fazendo uma comparação de desempenho entre eles.

REFERÊNCIAS

(ALLDAS, 2001) IT-SECURITY INFORMATION NETWORK. Disponível em: <http:// www.alldas.de>. Acesso em 10 agosto 2001.

(ALLDAS, 2002) IT-SECURITY INFORMATION NETWORK. Disponível em: < http://defaced.alldas.org/?archives=os >. Acesso em 10 jan. 2002.

(ANDERSON, 1980) ANDERSON, James P. **Computer Security Threat Monitoring and Surveillance**, Technical Report, James P. Anderson Co., Fort Washington, PA, abr. 1980.

(ASAKA et al, 1999) ASAKA, M. et al. **A Method of Tracing Intruders by Use of Mobile Agents**. INET'99, jun 1999.

(AUER, 1995) AUER, Karl. **Agents**. Disponível em: <http://www.pcug.org.au/~kauer/ project/main.htm>. Acesso em 10 set 2001.

(BARRUS e 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, jun. 1998.

(BAUER e KOBLENTZ, 1988) BAUER, David S.; KOBLENTZ, Michael E. **NIDX**: An Expert System for Real-Time Network Intrusion Detection, Proceedings of the Computer Networking Symposium, pp. 90-106, Washington, DC, abr. 1988.

(BELGRAVE, 1995) BELGRAVE, Marc. **The Unified Agent Architecture**. A White Paper. Disponível em: <http://www.ee.mcgill.ca/~belmarc/uaa_paper.html>. Acesso em 20 mar 2001

(BERNARDES, 1999) BERNARDES, M.C. **Avaliação do uso de Agentes Móveis em Segurança Computacional**. dez. 1999. Dissertação (Mestrado em Ciência da Computação) - ICMC/USP, São Paulo.

(BONIFÁCIO Jr. et al, 1998) 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.

(BOOCH et al., 1999) BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling language user guide**. Addison-Wesley, Longman. 1999.

(BROOKS, 1991a) BROOKS, Rodney A. **Intelligence Without Reason,** Technical Report MIT AI Lab Memo 1293, Massachusetts Institute of Technology – Artificial Intelligence Laboratory, abr. 1991. Prepared for Computer and Thought, IJCAI-91. Disponível em: <http://www.ai.mit.edu/people/brooks/papers.html.>. Acesso em 12 fev. 2001.

(BROOKS, 1991b) BROOKS, Rodney A. **Intelligence Without Representation,** *Artificial Intelligence Journal*, 47:139-159, ago. 1991.

(BROOKS, 1991c) BROOKS, R. A. **New Approaches to Robotics,** *Science*, 253:1227-1232, set. 1991.

(C2, 1985) TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA**.** Washington, D.C: DOD 5200-28-STD Report - Department of Defense, 1985.

(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 Applications, ICCIMA'97, Gold Coast, Australia, p.276- 280, fev. 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, abr. 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.

(CENTRAX, 1999) CENTRAX CORPORATION. **Security Audit Training and Laboratory Support Final Report.** USA: Contract Number: GS-35F-4571G, 1999.

(CERT, 2000) ALLEN, J. et al, **State of the Practice of Intrusion Detection Technologies.** Pittsburgh Carnegie Mellon Univ. - Tech Report CMU/SEI-99-TR-028, 2000. Disponível em.< http://www.cert.org/>. Acesso em 22 fev. 2000.

(CERT, 2000a) CERT - COMPUTER EMERGENCY RESPONSE TEAM. **Defending Yourself:** The Role of Intrusion Detection Systems. Disponível em: http://www.cert.org/. Acesso em 22 fev. 2000.

(CERT, 2002) CERT - COMPUTER EMERGENCY RESPONSE TEAM. Disponível em: <http://www.cert.org/stats/cert_stats.html>. Acesso em 20 fev. 2001.

(CG, 2002) COMITÊ GESTOR DA INTERNET NO BRASIL. Disponível em: <http://www.cg.org.br/indicadores/brasil-mundo.htm>. Acesso em 12 fev. 2002.

(COLLINS e NDUMU, 1999a) COLLINS, J.; NDUMU, D. **The Zeus Agent Building Toolkit, Zeus Methodology Documentation**, The Application Realisation Guide, vol 3, Inteligent Systems Research Group, BT Labs, v 1.01, set. 1999.

(COLLINS e NDUMU, 1999b) COLLINS, J.; NDUMU, D. **The Zeus Agent Building Toolkit, Zeus Technical Manual**, Inteligent Systems Research Group, BT Labs, v. 1.0, set. 1999.

(COLLINS e NDUMU, 1999c) COLLINS, J.; NDUMU, D. **The Zeus Agent Building Toolkit, Zeus Realisation Guide**, The Modelling Guide, vol. 2, Inteligent Systems Research Group, BT Labs, v 1.01, set. 1999.

(CROSBIE e SPAFFORD, 1995) 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 em: <http://www.cs.purdue.edu/homes/spaf/tech-reps/9522.os>. Acesso em 13 set. 2000.

(CROSBIE e SPAFFORD, 1995a) 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 em: <http://www.cs.purdue .edu/homes/spaf/tech-reps/9508.os>. Acesso em 19 out. 2000.

(CSI, 2001) COMPUTER SECURITY INSTITUTE. **Computer Crime and Security Survey.** Disponível em: <http://modulo.com.br>. 02 maio 2001.

(CHAIB-DRAA, 1994) CHAIB-DRAA, B. **Distributed artificial intelligence**: An overview. In A. Kent and J. Williams, editors, Encyclopedia of Computer Science and Technology, vol. 31 (suppl. 16), pages 215-243. Marcel Dekker, INC, 1994. Disponível em: <http://iris.ift.ulaval.ca/publications/chaib/pub-95/pub-95.html>. Acesso em 05 out. 2000.

(CHAIB-DRAA, 1995) CHAIB-DRAA, B. **Industrial applications in distributed ai**. Comunications of the ACM, 38 (11):49-53, nov. 1995. Disponível em: <http://iris.ift.ulaval.ca/publications/chaib/pub-95/pub-95.html>. Acesso em 05 out. 2000.

(DAVIS e SMITH, 1983) DAVIS, R.; SMITH, R.G. **Negotiation as a metaphor for distributed problem solving**. Artificial Intelligence 20, p. 63–109, 1983.

(DEITEL, 2001) DEITEL, H.M.; DEITEL, P.J. **Java, como programar.** Porto Alegre: Bookman, 2001.

(DENNING, 1987) DENNING, Dorothy E. **An Intrusion Detection Model**, IEEE Transactions on Software Engineering, Vol. SE-13, No. 2, pp. 222-232, fev. 1987.

(DIAS, 2001) DIAS, Rômulo Alves. **Agentes de Monitoramento e Avaliação de Segurança para o NIDIA**. Dissertação (Mestrado em Ciência da Computação) - UFMA, Maranhão (em fase de elaboração).

(DIENG, 1990) DIENG, Rose. **Méthodes et outils d'acquisition des connaissances**. *Er-gol*A, 1990.

(DOEDERLEIN, 1999) DOEDERLEIN, Osvaldo Pinali. **The Java Performance Report**. Disponível em: <http://www.javalobby.org/features/jpr/>. Acesso em 02 mar 2000.

(DURFEE et al., 1989) DURFEE, E. H.; LESER, V. R.; CORKILL, D. D. **Trends in cooperative distributed problem solving**. IEEE Trans. Knowl. Data Eng. – KOE, 11(1):63-83, 1989.

(DURFEE et al., 1994) DURFEE, E. H.; GMYTRASIEWICZ, P. J.; ROSENSCHEIN, J. S. **The utility of embedded communications**: Toward the emergence of protocols. In In Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop, p. 85-93, jul 1994. Disponível em: <http://ai.eecs.umich.edu/diag/homepage.html>. Acesso em 07 maio 2001.

(FERBER e GHALLAB, 1988) FERBER, J.; GHALLAB, M. **Problématiques des univers multi-agents intelligents**. Actes des journées nationeles PRC-GRECO Intelligence Artificialle, p. 295-320. Toulouse. Mars 1988.

(FININ et al., 1997) FININ T.; LABROU Y.; MAYFIELD, J. **KQML as an agent communication language**, Bradshaw J. ed., Sofware agents, MIT Press, Cambridge. 1997.

(FIPA, 1997a) FIPA. **Agent communication language**. Technical report. Foundation for Intelligent Physical Agents. 1997.

(FIPA, 1997b) THE FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS. **The FIPA'97 Specification**, 1997. Disponível em: http://drogo.cselt.it/fipa/spec/fipa97/fipa97.htm>. Acesso em jun. 2001.

(FRANKLIN e GRAESSER, 1996) FRANKLIN, Stan; GRAESSER. **Is it an Agent, or just a Program?**: A Taxonomy for Autonomous Agents. Disponível em: <http://www.msci.memphis.edu/~franklin/AgentProg.html>. 1996.

(FRINCKE et al., 1998) FRINCKE, D.; TOBIN, Don; MCCONNELL, Jesse; MARCONI, Jamie; POLLA, Dean. **A Framework for Cooperative Intrusion Detection**. Proceedings of the 21 st National Information Systems Security Conference, pp. 361-373, October 1998. Disponível em: <http://csrc.nist.gov/nissc/1998/papers.html>. Acesso em 60 jul. 2000.

(GARVEY e LUNT, 1991) GARVEY,T. D.; LUNT, T. F. **Model based Intrusion Detection**. USA: 14th National Computer Security Conference, 1991.

(GASSER e HUHNS, 1989) GASSER, M. N.; HUHNS, L. **Themes in distributed artificial intelligence research**. In Les Gasser and Michel N. Huhns, editors, Distributed Artificial Intelligence, pages 3-36. 2. vol. Pitman Publishing, 1989.

(GENESERETH e FIKES, 1992) GENESERETH, M.R.; FIKES, R.E. (1992) (Eds), **Knowledge Interchange Format**, version 3.0 Reference Manual, Computer Science Department, Stanford University, Technical Report Logic-92-1. Disponível em: <http://www-ksl.stanford.edu/knowledge-sharing/kif>. Acesso em 10 set. 2001.

(GILBERT et al., 1996) GILBERT, Don; APARÍCIO; MANNY, A. **Intelligent agent strategic**. Disponível em: <http://activist.gpl.ibm.com:81/WhitePaper/ptc2.html>. Acesso em 12 jul 2001.

(GOSLING et al., 1997) GOSLING, J.; JOY, B.; STEELE, G. **The java language specification**, Addison Wesley. 1997.

(GRUBER, 1993) GRUBER, T. **A translation approach to portable ontology specifications**. Knowledge Acquisition, 5(2), p. 199–220, 1993.

(HALPERN e MOSES, 1984) HALPERN, J.Y.; MOSES, Y. **Knowledge and Common Knowledge in a Distributed Environment**. In 3rd ACM Conference Principles of Distributed Computing, p. 50-61, 1984.

(HELMER et al., 1998) HELMER, Guy; WONG, Johnny S. K.; HONAVAR, Vasant; MILLER, Les. **Intelligent Agents for Intrusion Detection**. Proceedings, IEEE Information Technology Conference, Syracuse, NY, pp. 121-124, September 1998. Disponível em: <http://www.cs.iastate.edu/~ghelmer/phd.html>. Acesso em 04 abr. 2000.

(HOGG e HUBERMAN, 1991) HOGG, B. A.; HUBERMAN, T. **Controlling chaos in distributed systems**. IEEE Transactions on Systems, Man and Cybernetics, 21 96):1325-1332, nov. 1991.

(ISS, 2001) INTERNET SECURITY SYSTEMS. Disponível em: <http://www.iss.net/xforce>. Acesso em 21 set. 2001.

(JACOBSON et al., 1999) JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The unified software development process**. Addison-Wesley, Longman. 1999.

(JAVA, 2000) SUN MICROSYSTEMS. **Java 2 SDK, Standard Edition Version 1.2 Software**. Disponível em: < http://java.sun.com/products/jdk/1.2/>. Acesso em 11 nov. 2000.

(JENNINGS, 1996) JENNINGS, N. R. **Coordenation techniques for distributed artificial intelligence.** In G. M. P. O´Hare and N. R. Jennings, editors, Foundations of Distributed Artificial Intelligence, pages 187-210. Wiley, 1996. Disponível em: <http://www.elec.qmw.ac.uk/dai/publications/#1996>. Acesso em 22 ago. 2001.

(FOX et al., 1990) FOX, Kevin L.; HENNING , Ronda R.; REED, Jonathan H.; SIMONIAN, Richard. **A Neural Network Approach Towards Intrusion Detection**. In Proceedings of the 13th National Computer Security Conference, p. 125 -134, Washington, DC, out. 1990.

(FUJII, 2000) FUJII, Keita. **Java package for libpcap**. Disponível em: <http://www.goto.info.waseda.ac.jp/~fujii/jpcap/>. Acesso em 04 dez. 2000.

(KLEIN, 1991) KLEIN, Mark. **Supporting conflict resolution in cooperative design systems**. IEEE Transactions on Systems, Man and Cybernetics, 21(6):1379-1390, nov. 1991.

(KREMER, 1999) KREMER, Steven. **Real-time Intrusion Detection for Windows NT Based on Navy IT-21 Audit Policy**. Monterey, 1999. Dissertação (Mestrado em Computação), Naval Postgraduate School, 1999.

(KUMAR, 1995) KUMAR, Sandeep. **Classification and Detection of Computer Intrusions**. USA, 1995. Tese (Doutorado em Filosofia), Purdue University, 1995.

(LABIDI et al., 1993) LABIDI, S. et al. Lejouad, W. **De l´inteligence artificialle distribuée aux systèmes multi-agents**. Rapport de recherche INRIA. Nº 2004. France, ago. 1993.

(LEE et al., 1999) LEE, W.; STOLFO, S.J.; MOK, K. **A Data Mining Framework for Building Intrusion Detection Models**. Proceedings of the IEEE Symposium on Security and Privacy, 1999. Disponível em: <http://www.cs.columbia.edu/~sal/JAM/PROJECT/>. Acesso em 12 abr. 2000.

(LIZOTTE e MOULIN, 1990) LIZOTTE, M; MOULIN, B. **A Temporal Planner for Modelling Autonomous Agents**. In: Decentralized A.I. 1990.

(LUCK e D'INVERNO, 1995) LUCK, Michael; D'INVERNO, Mark. **A Formal Framework for Agency and Autonomy** In: ICMAS-95*,* 1995.

(LUNT, 1993) LUNT, Teresa F. **A Survey of Intrusion Detection Techniques**. Computerse Security, USA, v.12,n.04, p. 405-418, jun. 1993.

(Lunt et. al., 1992) LUNT , T. F.; TAMARU, A.; GILHAM, F.; JAGANNATHAN, R.; NEUMANN , P. G.; JAVITZ, H. S.; VALDES, A.; GARVEY, T. D. **A Real-Time Intrusion Detection Expert System (IDES)**. Final Technical Report. Computer Science Laboratory, SRI International, Menlo Park, California, fev. 1992.

(MAES, 1995) MAES, Pattie. **Artificial Life Meets Entertainment: Life like Autonomous Agents**. USA: ACM, 1995.

(MARTIN et al., 1998) MARTIN, D.L.; CHEYER, A. J.; MORAN, D. B. **The Open Agent Architecture**: A Framework for Building Distributed Software Systems. Applied AI Journal, THIS ISSUE. 1998.

(MCCLURE et al, 2000) MCCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. **Hackers Expostos**: Segredos e soluções para a segurança de redes. 2. ed. São Paulo: Makron Books, 2000. 469 p.

(MÓDULO, 2000) MÓDULO SECURITY SOLUTIONS. **6ª Pesquisa Nacional sobre Segurança da Informação**. Disponível em: <http://www.modulo.com.br>. Acesso em 25 abr. 2001.

(MÓDULO, 2001) MÓDULO SECURITY SOLUTIONS. **IDS-Peça chave em soluções de segurança**. Disponível em: <http://www.modulo.com.br>. Acesso em 25 abr. 2001.

(MÓDULO, 2001a) MÓDULO SECURITY SOLUTIONS. **7ª Pesquisa Nacional sobre Segurança da Informação**. Disponível em: <http://www.modulo.com.br>. Acesso em 20 nov. 2001.

(MÓDULO, 2002) MÓDULO SECURITY SOLUTIONS. **Incidentes de Segurança - 2000/2001**. Disponível em: <http://www.modulo.com.br>. Acesso em 27 mar. 2002.

(MÓDULO, 2002) MÓDULO SECURITY SOLUTIONS. **Falta de especialização ocasiona falhas de segurança no firewall**. Disponível em: <http://www.modulo.com.br>. Acesso em 27 mar. 2002.

(NAVIA, 2001) NAVIA, Jacob. **LCC-Win32**: a free compiler system for Windows. Disponível em: < http://www.cs.virginia.edu/~lcc-win32/>. Acesso em 19 jul. 2001.

(NEGROPONTE, 1997) NEGROPONTE, N. 1997. **Agents: From Direct Manipulation to Delegation**. In Software Agents, ed. J. M. Bradshaw. Menlo Park, Calif.: AAAI Press. 1997.

(NEWMAN, 1996) NEWMAN, A. et al. **Usando JAVA**. Rio de Janeiro: Campus, 1997.

(NT, 1999) MICROSOFT TRAiNING AND CERTIFICATION. **Administrando Windows NT Server e Workstation**, Argentina: Docuprint S. A., 1999.

(NT, 1999a) MICROSOFT TRAiNING AND CERTIFICATION. **Segurança no Windows NT Server e Workstation**, Argentina: Docuprint S. A., 1999.

(NWANA e WOOLDRIDGE, 1997) NWANA, H.S.; WOOLDRIDGE, M. **Software Agent Technologies**. in Nwana, H.S. e Azarmi, N. (eds.) Sof-ware Agents and Soft Computing: Concepts and Applications, Lecture Notes in Artificial Intelligence Series 1198, Berlin: Springer-Verlag, p. 59-78. 1997.

(NWANA et al., 1999) NWANA, H.; NDUMU, D.; LEE, L.; COLLINS, J. **ZEUS**: a toolkit for building distributed multi-agent systems. In Applied Artificial Intelligence Journal, Vol 13 (1), p. 129-186. 1999.

(ODELL et al., 2000) ODELL, J.; PARUNAK, H.V.D.; BAUER, B. **Extending UML for Agents**. Proc. Of the Agent-Oriented Information Systems Workshop at the 17th NationalConference on Artificial Intelligence, p. 3-17 accepted paper, AOIS Workshop at AAAI 2000.

(OLIVEIRA, 1996) OLIVEIRA, F. **Inteligência Artificial Distribuída**. In: IV Escola Regional de Informática, SBC, SC, 1996.

(OLIVEIRA, 2002) OLIVEIRA, Antonio Alfredo Pires. Proposta de um servidor armadilha, baseado em agentes inteligentes, para investigação de atos suspeitos em sistemas de computadores.

(OMG, 2000) AGENT WORKING GROUP. **Agent Technology**, Object Management Group, v. 1, 1 August 2000.

(PARAÍSO, 1996) PARAISO, Emerson C. **MASC - Sistema Multi-Agente para Monitoração e Controle de Processos**. Disponível em: <http://dainf.cefetpr.br/~paraiso/>. Acesso em 13 maio 2000.

(PORRAS e KEMMERER, 1992) PORRAS, Phillip A.; KEMMERER, Richard. A. **Penetration State Transition Analysis**: a Rule-Based Intrusion Detection Approach. USA: Eighth Annual Computer Security Applications Conference, 1992.

(PRNEWSWIRE, 1999) PR NEWSWIRE ASSOCIATION, Inc. **Plugging the Holes in eCommerce Leads to 135% Growth in the Intrusion Detection and Vulnerability Assessment Software Market**. PRNewswire. ago. 10, 1999.

(RANUM, 1992) RANUM, M. J.. **An Internet Firewall**, proceedings of World Conference on Systems Management and Security, 1992. Disponível em: <ftp://decuac.dec.com/pub/docs/firewallfirewall.ps>. Acesso em 12 jun. 2000.

(RAO e GEORGEFF, 1991) RAO, A.S.; GEORGEFF, M.P. **Modeling rational agents within a BDI-architecture**. In Fikes, R. and Sndewall, E., editors, Proceedings of Knowledge Representation and Reasoning (KReR-91), p. 473-484. Morgan Kaufmann Publishers: San Mateo, CA. 1991.

(RITCHEY, 1996) RITCHEY, Tim. **Programando com JAVA**. Rio de Janeiro: Campus, 1996.

(SANTOS, 2002) SANTOS, Glenda de Lourdes Ferreira. **Um Agente Inteligente Controlador de Ações do Sistema**. Dissertação (Mestrado em Ciência da Computação) - UFMA, Maranhão (em fase de elaboração).

(SEARLE, 1969) SEARLE, J.R. **Speech acts**. Cambridge University Press, Cambridge MA, 1969.

(SEARLE, 1983) SEARLE, J. R. **Intentionality***:* An Essay in the Philosophy of Mind. Cambrige University Press, New York, 1983.

(SFOCUS, 2001) SECURITY FOCUS. Disponível em: <http://www.securityfocus.com/vulnerability>. Acesso em 21 set. 2001.

(SFOCUS, 2001a) SECURITY FOCUS. Disponível em: <http://www.securityfocus.com/>. Acesso em 10 ago. 2000.

(SHOHAM, 1993) SHOHAM, Y. **Agent oriented programming. Artificial Intelligence**, 60(1):51-92. 1993.

(SMAHA, 1988) SMAHA, Stephen E., **Haystack**: An Intrusion Detection System. Fourth Aerospace Computer Security Applications Conference, Orlando Florida, p. 37-44, dez. 1988.

(SNAPP e SMAHA, 1992) SNAPP, Steven R.; SMAHA Stephen E. **Signature Analysis Model Def-inition and Formalism**. In Proceedings of the Fourth Workshop on Computer Security Incident Handling, Denver, Colorado, ago. 1992.

(SOBIREY e RICHER, 1999) SOBIREY, M; RICHTER, B. **Funktionsbeschreibung des Intrusion-Detection-Systems AID**, Technischer Bericht E/I11S/X0034/Q5123 - I.a, 1999.

(SOUSA, 2002) SOUSA, Antonio Manoel Silveira de. **Classificação de ataques e ações para o SCA (Agente Controlador de Ações)**. Dissertação (Mestrado em Ciência da Computação) - UFMA, Maranhão (em fase de elaboração).

(STEELS, 1995) STEELS, Luc. **When are robots intelligent autonomous agents?** Journal of Robotics and Autonomous Systems, 15:3-9, 1995. Disponível em: <http://arti.vub.ac.be/www/steels/publications.html>. Acesso em 30 jun 2000.

(SUN, 2001) SUN MICROSYSTEMS, **JAVA (TM) 2SDK Documentation**. Disponível em: <http://java.sun.com/products/jdk/1.2/docs/>. Acesso em 07 jan. 2001.

(TENG e CHENG, 1990) TENG, H.S.; CHENG, K. **Security Audit Trail Analysis Using Inductively Generated Predictive Rules**. New Jersey: Sixth Conference on Artificial Intelligence Applications, 1990.

(WINPCAP, 2000) LAWRENCE BERKELEY LABORATORY. **Free Packet Capture Architecture for Windows**. Disponível em: < http://netgroup-serv.polito.it/winpcap/>. Acesso em 04 dez. 2000.

(WOOLDRIDGE e JENNINGS, 1995) WOOLDRIDGE, M.; JENNINGS, N. **Intelligent agents**: theory and practice . The Knowledge Engineering Review. 1995.

(ZAMBONI et. Al, 1998) BALASUBRAMANIYAN, Jai; GARCIA-FERNANDEZ, Jose Omar; ISACOFF, David; SPAFFORD, E. H.; ZAMBONI, Diego. **An Architecture for Intrusion Detection using Autonomous Agents**. Department of Computer Sciences, Purdue University, Coast TR 98-05, 1998. Disponível em: <http://www.cs.purdue.edu/coast/coast-library.html>. Acesso em 17 out. 2000.

(ZEUS, 2000) BT INTELLIGENT AGENT RESEARCH. **Zeus Toolkit**. Disponível em: < http://www.btexact.com/projects/agents/zeus/>. Acesso em 12 dez. 2000.

APÊNDICE A - MODELAGEM DE SISTEMAS MULTIAGENTES USANDO A PLATAFORMA ZEUS

1 AGENTES

Atualmente, a palavra “agente” tem tido grande visibilidade. Muitos pesquisadores têm desenvolvido trabalhos com base nesta tecnologia. Empresas de grande porte, como IBM, Mitsubishi e British Telecommunications têm vislumbrado nesta abordagem um forte potencial comercial e têm investido enorme quantidade de recursos financeiros (OMG, 2000).

Até o momento não há um consenso universal dos pesquisadores quanto à definição de agentes (LUCK e D'INVERNO, 1995). Isso ocorre devido a diversos fatores relacionados a complexidade dos termos utilizados (racionalidade, autonomia, etc), além da falta de coordenação entre as pesquisas realizadas ao longo dos anos (FRANKLIN e GRAESSER, 1996).

Segundo (FERBER e GHALLAB, 1988), um agente é definido como:

> *“[...] uma entidade (física ou abstrata), capaz de agir sobre ela mesma e sobre o seu ambiente, possuindo uma representação parcial deste ambiente, podendo comunicar-se com outros agentes, e cujo comportamento é conseqüência de suas observações, de seus conhecimentos e de suas interações com outros agentes”.*

Jennings e Wooldrigde propuseram uma lista de atributos essenciais (autonomia, habilidade social, reatividade e pró-atividade) e não-essenciais (mobilidade, veracidade e benevolência) e características aplicáveis aos humanos, os estados mentais e emocionais (conhecimento, crenças[1] e intenções, etc) (Rao

[1] Um conhecimento adquirido ao longo do tempo, conhecido como “incerto”, pode ser definido como uma crença de um agente. Em (Halpern e Moses, 1984), pode ser encontrado um estudo mais aprofundado sobre as crenças de um agente.

e Georgeff, 1991; Shoham, 1993), que podem ser utilizados para conceituar agente e agência[2] (WOOLDRIDGE e JENNINGS, 1995).

Maes (1995) define agentes inteligentes como sistemas computacionais residentes em ambientes dinâmicos complexos, os quais percebem e atuam autonomamente neste ambiente e, ao fazê-lo, realizam um conjunto de objetivos e tarefas para os quais foram designados. Por outro lado, (Gilbert, 1996) conceitua agentes inteligentes como sendo entidades de software que realizam um conjunto de operações em benefício do usuário ou de outro programa, com um certo grau de independência ou autonomia, empregando algum conhecimento ou representando os objetivos ou preferências do usuário.

Os agentes possuem dois aspectos: um social e um individual. O aspecto social está relacionado aos mecanismos e conhecimentos associados às atividades do grupo, enquanto que o aspecto individual está relacionado aos mecanismos e conhecimentos associados às regras de funcionamento interno do agente (STEELS, 1995).

Um agente é caracterizado através de seu papel, sua especialidade, seus objetivos, suas funcionalidades, suas crenças, sua capacidade de decisão, comunicação e aprendizagem (LABIDI et al., 1993).

De acordo com o grau de complexidade de suas funções (granularidade), um agente pode ser classificado como**: agente cognitivo** e **agente reativo**.

O agente reativo é visto como sendo o de mais baixo nível e sua capacidade limita-se a *situação/ação* ou *estímulo/resposta.* Esta abordagem é defendida pelo fato de que em um sistema multiagentes não é necessário que todos os agentes sejam inteligentes para que o comportamento global seja inteligente (Brooks, 1991a, 1991b, 1991c).

O agente cognitivo, com forte granularidade, é dotado da capacidade de raciocinar e aprender. Os sistemas cognitivos são baseados na cooperação de

[2] Conceitua-se agência como sendo um grupo de agentes realacionados.

agentes, com capacidade de operações complexas. Esses sistemas compreendem um pequeno número de agentes capazes de raciocinar sobre uma base de conhecimentos e de colaborar entre si para que objetivos comuns sejam alcançados (DIENG, 1990).

Um agente cognitivo é caracterizado através de sua:

i) ***Intencionalidade*** – capacidade de exprimir a vontade de alcançar um objetivo ou realizar uma ação (SEARLE, 1983)*;*

ii) ***racionalidade*** - capacidade de justificar suas ações e dispor de critérios para avaliá-las e selecioná-las para alcançar seus objetivos com sucesso (AUER, 1995)*;*

iii) ***comportamento colaborativo (cooperação)*** – capacidade de poder colaborar com outros agentes para que objetivos comuns sejam alcançados (GILBERT et al., 1996; PARAÍSO, 1996)*;*

iv) ***adaptabilidade*** – capacidade de aprender e melhorar ao longo do tempo para adaptar-se a diversos ambientes (FRANKLIN e GRAESSER, 1996; GILBERT et al., 1996)*;*

v) ***autonomia*** - capacidade do agente de executar o controle sobre suas próprias ações (FRANKLIN e GRAESSER, 1996);

vi) ***capacidade de fazer inferências*** - um agente pode aumentar o seu grau de flexibilidade, fazendo uso de suas crenças para alcançar seus objetivos (BELGRAVE, 1995). As crenças de um agente contribuem para o comportamento inteligente de um sistema multiagentes.

2 SISTEMAS MULTIAGENTES

2.1 Linguagem Java

Atualmente, várias instituições comerciais e de pesquisa (OMG, 2000) estão utilizando Java, linguagem de programação orientada a objeto da Sun Microsystems (Gosling et al., 1997), no desenvolvimento de seus sistemas multiagentes, devido a sua portabilidade, suporte à programação concorrente (*multithrea*d) e por possuir um ambiente de execução relativamente seguro (NEWMAN, 1996; RITCHEY, 1996; DEITEL, 2001).

A portabilidade é o maior benefício desta linguagem, que dispondo de uma arquitetura neutra, faz com que seja independente de plataforma, permitindo que o mesmo opere em ambientes heterogêneos. O compilador Java traduz os códigos fontes para um código intermediário e independente de plataforma, denominado *Java Byte Code*, que é interpretado por uma máquina virtual, denominada JVM (*Java Virtual Machine*), durante a execução. Todo sistema que suporta a JVM pode executar programas escritos em Java.

Uma característica importante é que, sendo uma linguagem projetada para trabalhar em ambientes distribuídos, dispõe de um mecanismo (o modelo de ponteiros Java) desenvolvido para inibir a ação de vírus, eliminando-se a possibilidade de sobrescrição de memória e corrompimento de dados. Ainda que os *byte codes* venham a ser corrompidos, ele dispõe de mecanismos de segurança que atuam, impedindo que a semântica básica da linguagem seja violada. Além disso, não permite tipos de dados ilegais ou ponteiros aritméticos e possui um gerenciador de segurança para checar todas as operações potencialmente inseguras, bem como o acesso a arquivos e conexões com a rede, para determinar se o programa tem permissão para executar tais operações.

Outras características que contribuem para o sucesso do uso da linguagem Java na implementação de agentes são: *i)* carregamento dinâmico de classes – a JVM carrega e define classes (locais ou remotas) em tempo de execução, fornecendo um espaço de nome protegido para cada agente e

permitindo, desta forma, sua execução segura e independente dos demais agentes; *ii)* programação concorrente, onde cada agente pode ser executado em uma linha de execução (*thread*) independentemente de outros agentes dentro do mesmo ambiente de execução, proporcionando autonomia aos agentes e melhorando a *performance* do sistema; *iii)* a serialização de objetos – muito importante no desenvolvimento de agentes móveis, faz com que os agentes tenham os valores dos seus atributos (estado dos objetos) armazenados de maneira serializadas, de forma que possam ser reconstituídos (desserializados) mais tarde, em uma nova instância.

Porém, apesar de possuir tantas vantagens, a linguagem Java possui algumas limitações (SUN, 2001), que não podem ser esquecidas ou negligenciadas, na hora da implementação:

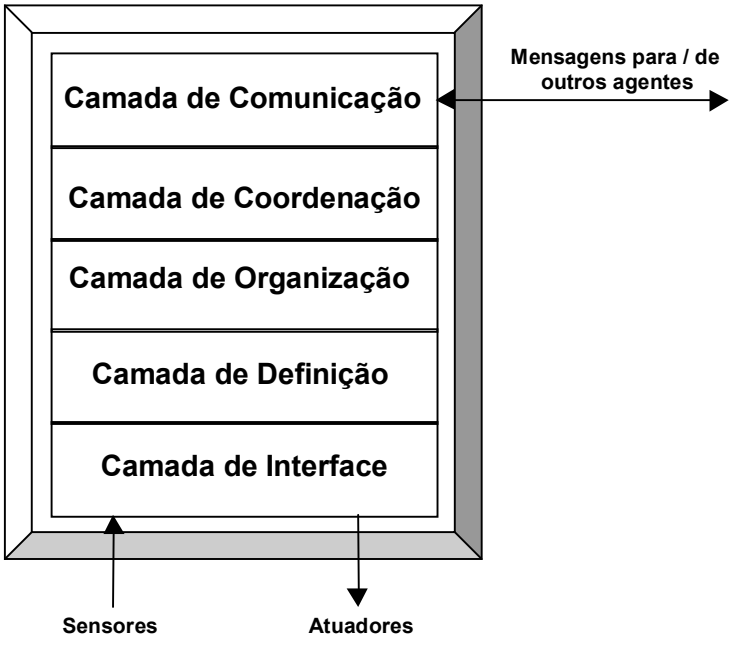
- não possui suporte adequado para controlar os recursos do sistema, ou seja, um agente pode alocar recursos (processador e memória) demasiados da máquina, podendo ocasionar uma negação de serviço (*Denial of Service*), ataque este largamente utilizado pelos *hackers*;
- não possui nenhuma proteção a referências, ou seja, um agente não dispõe de mecanismos para monitorar e controlar o acesso de outros agentes a seus métodos públicos e sendo assim, alguns objetos Java têm acesso a um número maior de métodos do que outros;
- não possui suporte para prevenção e recuperação do estado de execução de um objeto – característica importante na construção de agentes móveis – um agente perde todas as informações do *status* do contador de programas e das pilhas de execução quando é transferido de uma máquina para outra. Este problema pode ser amenizado utilizando-se a serialização de objetos.

2.2 Zeus – ferramenta de construção de agente

A plataforma ZEUS (Nwana et al., 1999), desenvolvida pelo Grupo de Pesquisa de Sistemas Inteligentes da *British Telecommunications* (Laboratórios BT), é uma síntese de tecnologias de agentes para gerar um ambiente integrado ao desenvolvimento rápido de sistemas multiagentes, que apresenta algumas características as quais são motivadoras quanto à sua escolha:

- é gratuita, possui código aberto e é escrita em Java;
- dispõe de uma interface gráfica para desenvolvimento de agentes;
- opera em redes TCP/IP, facilitando, com isso, a intercomunicação entre agentes que estão sendo executados em diferentes sistemas operacionais.

Em um alto nível de abstração, um agente ZEUS pode ser visto como uma entidade composta por cinco camadas conceituais (conforme ilustrado na Figura A.1).



Fonte: (COLLINS e NDUMU, 1999a)

Figura A.1 - Estrutura Conceitual de um Agente ZEUS (adaptação da autora).

i) **camada de interface** – que habilita o agente a ser interligado a um programa externo, fornecendo-lhe recursos e/ou implementando suas competências. Esta camada recebe a informações de entrada de seu meio ambiente através de seus sensores e age sobre o mesmo através de seus atuadores.

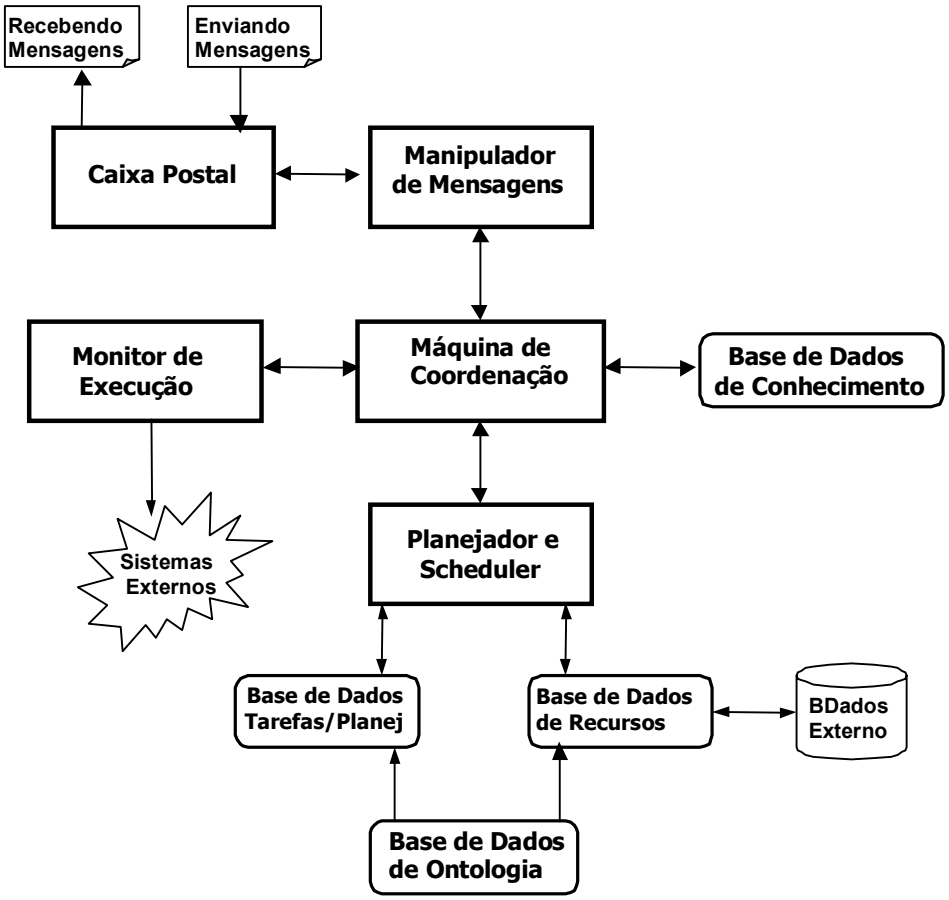
ii) **camada de definição** – onde o agente é visto como uma entidade autônoma, baseada em suas habilidades, metas, recursos, crenças, racionalidade e preferências.

iii) **camada de organização** – onde o agente é visto em termos de seu relacionamento com outros agentes (agência).

iv) **camada de coordenação** – onde o agente é visto como uma entidade social que interage de acordo como seus protocolos e suas estratégias.

v) **camada de comunicação** – que implementa os protocolos e mecanismos que suportam a comunicação inter-agentes.

Um agente ZEUS pode ser configurado para um domínio específico de aplicação através da inclusão de recursos específicos do problema, competências, informação, relacionamentos organizacionais e protocolos de coordenação. A arquitetura genérica do Agente ZEUS é mostrada na Figura A.2.



Fonte: (COLLINS e NDUMU, 1999b)

Figura A.2 – Arquitetura genérica do Agente ZEUS (adaptação da autora).

i) **Caixa Postal (*Mailbox*)** – que manipula a comunicação entre os agentes, proporcionando-lhes a habilidade para enviar e receber mensagens em um formato padrão.

ii) **Manipulador de Mensagem** – que processa as mensagens recebidas pela Caixa postal e as despacha para os componentes pertinentes do agente. Seu comportamento é controlado através de dois aspectos: primeiro, se uma nova mensagem representa o início de um novo diálogo ou é parte de um diálogo existente; e

segundo, nas regras de processamento da mensagem registradas no Manipulador por outros componentes do agente.

iii) **Máquina de Coordenação** – que toma decisões relativas aos objetivos do agente e é também responsável pela coordenação das suas interações com outros agentes que usam os mesmos protocolos de coordenação e estratégias.

iv) **Banco de Dados de Conhecimento** – que descreve as relações do agente com os outros na sociedade e suas crenças sobre as capacidades desses agentes. A Máquina de Coordenação usa as informações desta base de dados para permitir a colaboração com outros agentes.

v) **Planejador/*Scheduler*** – que planeja e agenda as tarefas do agente baseado nas decisões informadas pela Máquina de Coordenação, nos recursos e especificações das tarefas disponíveis para o agente.

vi) **Banco de Dados de Recursos** – que mantém uma lista de recursos (fatos) próprios e disponíveis ao agente. Esta base de dados suporta uma interface direta para sistemas externos, que permitem sua interligação dinamicamente e a utilização de bancos de dados proprietários.

vii) **Banco de Dados de Ontologia** – que armazena a definição lógica de cada tipo de fato – seus atributos legais; a faixa de valores legais para cada atributo; qualquer constrangimento entre os valores do atributo e qualquer relação entre os atributos do fato e os outros fatos. O uso desta base de dados facilita a comunicação entre os agentes.

viii) **Banco de Dados de Tarefa/Planejamento (*Task/Plan*)** – que fornece descrições lógicas dos operadores de planejamento (ou tarefas) conhecidos para o agente.

ix) **Monitor de Execução** – que mantém o relógio interno do agente, o início, as paradas e as tarefas dos monitores que foram agendadas para execução ou término pelo Planejador/*Scheduler*. Também informa ao Planejador das condições de término (com sucesso ou excepcionais) das tarefas gerenciadas.

O Monitor de Execução dispõe de uma interface direta para administrar sistemas externos. Isso implica na possibilidade do agente utilizar bancos de dados proprietários, bem como executar programas que realizem tarefas para solucionar problemas de domínio específico.

Um fato importante do uso dessa característica é que o código externo pode utilizar os métodos públicos do agente para solicitar ou modificar o seu estado interno. Com isso, por exemplo, os recursos e/ou bases de dados de planejamento podem ser solicitados ou modificados.

Desta forma, uma vez que todas as funcionalidades de comunicação inter-agentes, planejamento e execução de tarefas da sociedade já estão prontas no ambiente, a implementação passa a estar concentrada no problema específico (tarefas), desprezando preocupações extras com a criação e comunicação de agentes.

2.3 Comunicação entre agentes

A cooperatividade evidenciada na IAD faz com que mecanismos de comunicação sejam necessários entre agentes de uma sociedade. Os agentes precisam se comunicar, visando seus objetivos locais ou, ainda, convergindo para um objetivo global.

A comunicação entre agentes esclarece as intenções individuais perante o contexto social, bem como a existência de coordenação nas atividades. O processo da comunicação em IAD, normalmente, é realizado por um esquema de protocolos (DURFEE et al., 1994).

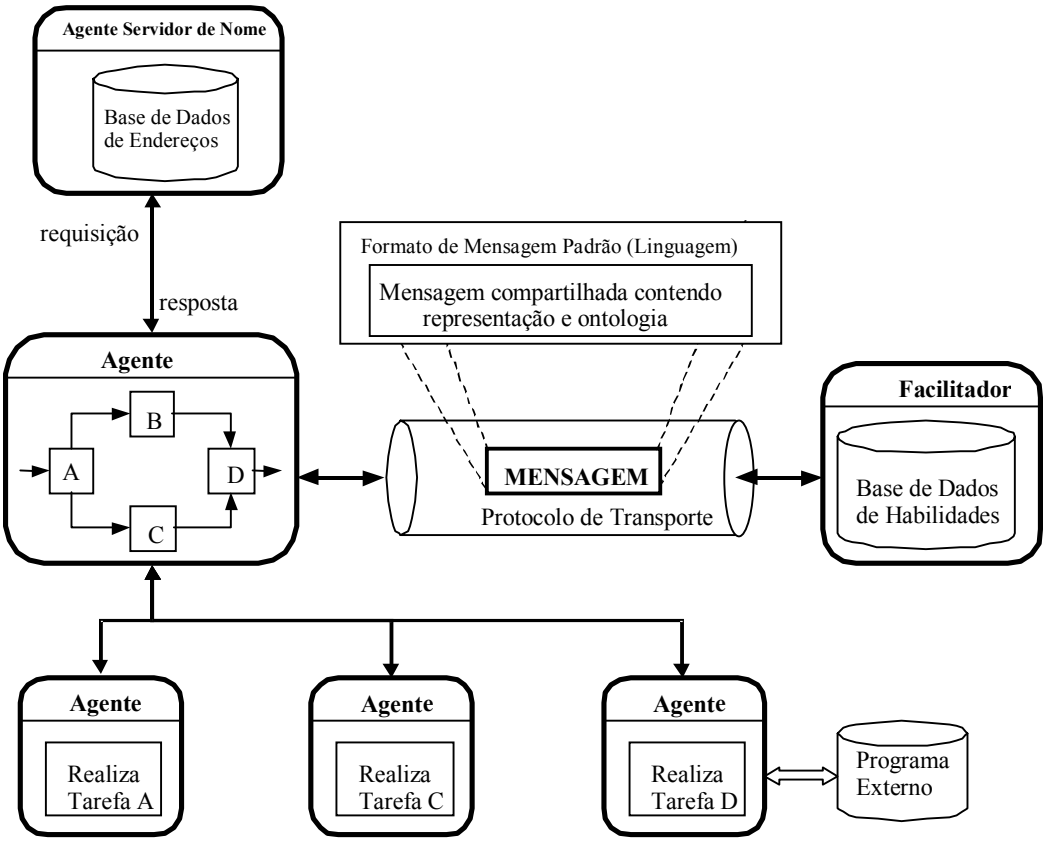
Protocolo é um conjunto de regras que normatiza e controla a troca de informações entre os agentes. O agente deve reconhecer qual o protocolo estabelecido pela sociedade para que esteja habilitado em interagir com os demais agentes. Um protocolo torna possível a participação social por parte de um agente, permitindo que efetue restrições e contribuições à sociedade ou, ainda, enriquecendo a si próprio (JENNINGS, 1996). Enfim, o uso de protocolos é oportuno para integrar os conhecimentos heterogêneos dos agentes, bem como torná-los disponíveis à sociedade.

De uma maneira geral, quando um agente necessita executar uma tarefa complexa que exige a colaboração de outros agentes, este usa o **Facilitator** para descobrir quais possuem as habilidades exigidas e o **Agente Servidor de Nome** para determinar os seus endereços (MARTIN et al., 1998). A linguagem de comunicação inter-agente é usada na comunicação com o Agente Servidor de Nome, o Facilitador e os demais agentes. Esta comunicação requer uma representação compartilhada e conhecida dos conceitos de domínio comuns (ontologia[3]) (Figura A.3).

Na plataforma ZEUS, o Agente Servidor de Nomes, o Facilitador e o Visualizador são conhecidos como agentes utilitários. Uma sociedade de agentes ZEUS pode conter vários agentes utilitários, com pelo menos um Agente Servidor de Nomes.

Agentes Servidores de Nomes possuem somente dois componentes - uma Caixa Postal e um Manipulador de Mensagens - necessários para receber e responder para os agentes as requisições de endereços de outros agentes. É também função deste agente manter um relógio que será utilizado para sincronizar os relógios dos agentes, inclusive dos agentes utilitários, no momento em que são inicializados na sociedade e registrados no servidor de nomes.

[3] Agentes que se comunicam em uma linguagem comum, ainda estarão impossibilitados de entender um ao outro, principalmente, se forem usados vocabulários diferentes para a representação de conceitos de domínio compartilhado. Eles também precisam usar a mesma ontologia ou vocabulário de conceitos comuns (GRUBER, 1993).

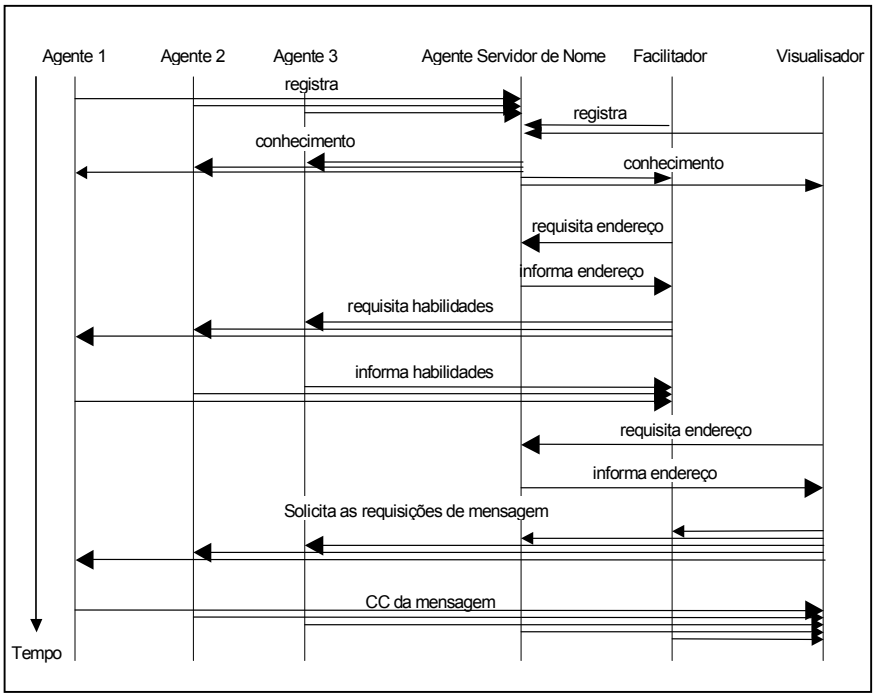


Fonte: (COLLINS e NDUMU, 1999b)

Figura A.3 – Diagrama de contexto da colaboração de um sistema multiagentes (adaptação da autora).

Os Agentes Facilitadores dispõem de uma Caixa Postal e um Manipulador de Mensagens para receber e responder para os outros agentes os seus questionamentos sobre as habilidades dos outros e uma Base de Dados de Conhecimento para salvar estas informações. Eles funcionam periodicamente perguntando a todos os agentes na sociedade sobre suas habilidades e armazenando tais informações em sua Base de Dados de Conhecimento. Também, agentes individuais podem anunciar suas habilidades aos Facilitadores. Assim, quando um agente deseja encontrar outros agentes que têm uma competência particular, eles podem simplesmente enviar uma mensagem de pergunta apropriada para um agente facilitador. O diagrama na Figura A.4

demonstra todas as interações ocorridas em um sistema multiagentes quando inicializado, utilizando mecanismos padrões ZEUS de passagem de mensagem.



Fonte: (COLLINS e NDUMU, 1999b)

Figura A.4: Diagrama de interação de uma sociedade de agentes (adaptação da autora)

Os Agentes Visualizadores podem ser utilizados para ver, analisar ou realizar o *debug* da sociedades de agentes ZEUS. Eles funcionam através de questionamentos a outros agentes sobre seus estados e processos e então coletam e interpretam as respostas para criar um modelo atualizado do comportamento coletivo dos agentes.

Algumas linguagens têm sido desenvolvidas para fornecer a Linguagem de Comunicação de Agentes (ACL) (NWANA e WOOLDRIDGE, 1997). Dentre elas, pode-se destacar KQML (*Knowledge Query and Manipulation Language*) (Finin et al., 1997) e FIPA ACL (FIPA, 1997a).

A maioria das linguagens de comunicação de agente está baseada na linguagem natural (Searle, 1969), em que são usadas expressões vocais de ações semelhantes às realizadas no mundo físico cotidiano.

Numa comunicação de linguagem natural, mesmo para simples tipos de diálogos, há o uso do conceito de estados mentais, como crenças, fatos, intenções e bases experimentais de conhecimento. Conseqüentemente, ACL´s especificam tipos de mensagens performativas (*performative*), tais como “pergunte”, “conte”, ou “alcance”.

Com essa formulação, a compreensão de diálogos é factível mesmo com a ausência de algumas palavras, uma vez que se conhece as bases do diálogo e suas intenções. De fato, diálogos entre agentes permitem a geração e a interpretação de declarações (verbalizações, opiniões, discursos, asserções, etc), que são ações faladas e planejadas, definindo um estado mental para o emissor (crenças, comprometimentos e intenções). Esses estados consistem em induzir um estado mental particular do receptor. “*Através de um diálogo apropriado, os agentes podem convergir sobre planos mentais compartilhados, (isto é, redes de crenças, comprometimentos, e intenções), pois dessa maneira eles coordenariam suas atividades*” (CHAIB-DRAA, 1994).

Em razão de que domínios de aplicação diferentes podem requerer linguagens de conteúdos diferentes, a maioria das ACL´s não especifica uma sintaxe ou semântica nos conteúdos das mensagens. Portanto, várias abordagens foram desenvolvidas, por exemplo KIF (*Knowledge Interchange Format*) (Genesereth e Fikes, 1992), tipicamente usado com KQML e FIPA SL (FIPA, 1997b), linguagem de conteúdo preferida para uso com o FIPA ACL.

Sob a ótica deste trabalho, uma sociedade de agentes ZEUS se comunica usando mensagens que obedecem a especificação FIPA 1997 ACL. A comunicação entre os agentes ZEUS é feita ponto a ponto, através de *sockets* TCP/IP, com cada mensagem enviada como uma seqüência de caracteres ASCII, usando a Caixa Postal e o Manipulador de Mensagem do agente e sendo tratadas usando estratégias de prioridade do tipo FIFO (*First In First Out*). A sintaxe da

linguagem está incluída na plataforma ZEUS que é usada pelo *parser* interno do agente para formular e decodificar as mensagens. Esta sintaxe é usada para construir instâncias da classe *performative*, que tem os atributos apresentados na Tabela A.1.

Fonte: (COLLINS e NDUMU, 1999b)

Tabela A.1 – Atributos da classe *Performative* (adaptação da autora)

```
Performative(
        type:           /* tipo performative, p. ex.. inform, cancel etc. */
        sender:         /* nome do agente que está enviando a mensagem */
        receiver:       /* nome do agente que receberá a mensagem*/
        reply_with:     /* chave de identificação da conversação do remetente */
        in_reply_to:    /* chave de identificação da conversação do receptor  */
        content:        /* conteúdo da mensagem */
        language:       /* nome da linguagem na qual o conteúdo está expressado */
        address:        /* endereço do remetente */
        send_time:      /* hora de envio da mensagem */
        receive_time:   /* hora de recebimento da mensagem */
)
```

O primeiro campo da mensagem – *type* – refere-se a ação que o agente receptor está sendo solicitado para executar, conforme a Tabela A.2.

As mensagens são originadas dos estados dos nós de diálogo ou do disparo de regras que têm um efeito de comunicação. Ao serem recebidas, tais mensagens são então processadas pelo módulo Manipulador de Mensagens. Um cenário demonstrativo será descrito na Seção seguinte.

Fonte: (COLLINS e NDUMU, 1999b)

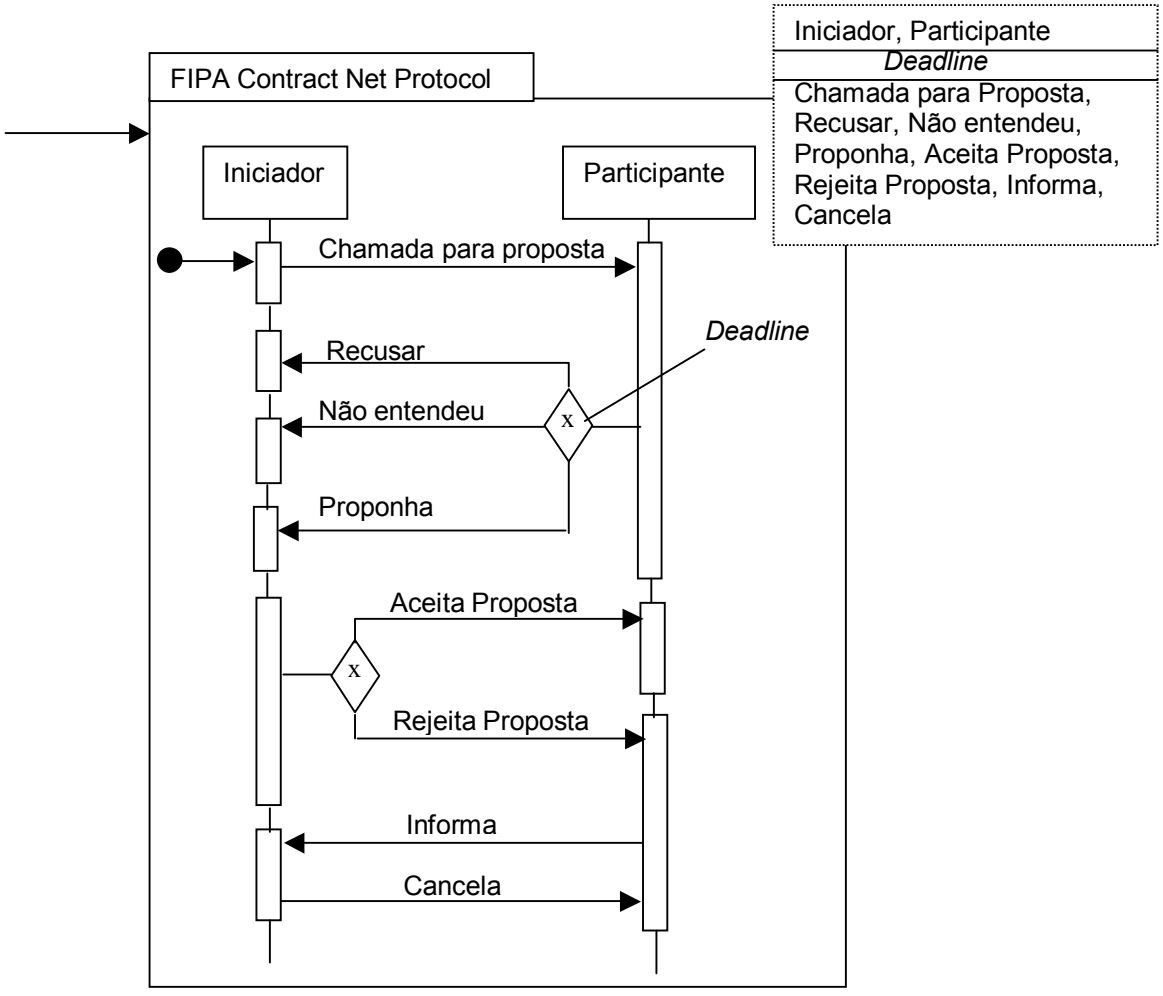
Tabela A.2: Os tipos de mensagens presentes no ZEUS (adaptação da autora).

Tipo de Mensagem	Finalidade
accept-proposal	Usado no contexto de um diálogo existente para informar ao receptor que uma proposta anterior foi aceita
agree	Sinal de concordância dos fatos acertados dentro da mensagem
cancel	Faz a Caixa Postal do receptor e o Monitor de Execução deixarem de se reportar ao remetente
cfp	Convida o receptor a fazer uma oferta relativo a um recurso especificado
confirm	Confirma a reserva de um recurso
disconfirm	Retrata um confirmação previamente feita
failure	Termina um diálogo quando o originador já não pode continuar
inform	Induz o receptor a adicionar o conteúdo associado ao seu Banco de Dados de Recurso
inform-if	Uma versão condicional de *Inform*
inform-ref	Induz o recipiente a adicionar o conteúdo referenciado ao seu Banco de Dados de Recurso
not-understood	Um caso mais específico de falha
propose	Encapsula uma proposta por alcançar ou comprar um recurso especificado
query-if	Um pedido condicional para informação
query-ref	Um pedido de informação referenciada pelo conteúdo da mensagem
refuse	Termina um diálogo, normalmente porque o originador considera que as propostas recebidas foram inaceitáveis (cf. *Failure*)
reject-proposal	Usado no contexto de um diálogo existente para informar ao receptor que uma proposta anterior é inaceitável
request	Pede para o receptor que forneça algum recurso ou serviço
request-when	Uma versão condicional do *Request*
request-whenever	Uma versão condicional de tempo do *Request*
subscribe	Registra o originador, tipicamente usado para registrar em agentes utilitários tais como os Servidores de Nomes e Visualisadores

2.4 Coordenação e cooperação

O comportamento coordenado dos sistemas multiagentes é uma área de pesquisa ativa com muitas técnicas em uso. As principais abordagens podem ser classificadas amplamente como estruturação organizacional, contratação, planejamento multiagentes e negociação. Na estruturação organizacional, a estrutura da sociedade definida como prioritária (isto é, os papéis diferentes dos agentes e suas relações com os demais) é explorada pela coordenação, como simbolizada através de sistemas cliente-servidor.

A contratação como um mecanismo de coordenação é simbolizado pelo protocolo de rede contratual clássico (Davis e Smith, 1983), onde um agente gerenciador anuncia um contrato, recebe as ofertas de outros agentes interessados e após serem avaliadas, o contrato é entregue ao agente premiado (Figura A.5). Esta é a abordagem utilizada pelo agente ZEUS, onde um ou mais *Iniciadores* emitem uma chamada para propostas (*cfp*), e um ou mais *Participantes* respondem à solicitação.



Fonte: (Odell et al., 2000)

Figura A.5 – Mecanismo de contratação (adaptação da autora)

No planejamento multiagentes, os agentes podem utilizar técnicas de planejamento da IA clássica para planejarem suas atividades e solucionarem qualquer conflito previsto. O planejamento regularmente leva a uma das duas formas: planejamento centralizado – na qual um agente central executa o planejamento em nome da sociedade; o planejamento descentralizado - onde os agentes trocam subplanos parciais e elaboram o plano global, solucionando, assim, os conflitos progressivamente.

Na negociação, os agentes empenham-se no diálogo, trocando propostas entre si, avaliando as propostas dos outros e modificando as suas próprias propostas até chegarem a um estado quando todos estão satisfeitos. Os mecanismos de negociação típicos estão baseados na teoria do jogo, em alguma forma de planejamento ou inspirada em negociações humanas.

De fato, a coordenação em um sistema multiagentes é um aspecto crucial. Sem ela, qualquer benefício de interação inexiste e o comportamento do grupo de agentes pode tornar-se caótico (CHAIB-DRAA, 1995).

A cooperação ocorre quando dois ou mais agentes operam de modo paralelo e concorrente, com um objetivo de promover uma ativação de um outro agente ou para atingir um objetivo global. A cooperação tem objetivos fundamentais na IAD, tais como (DURFEE et al., 1989):

i) aumento da taxa de realização de tarefas através do paralelismo;

ii) aumento do conunto ou do escopo de tarefas executáveis pelo compartilhamento de recursos;

iii) aumento das chances de finalização das tarefas pelo engajamento redundante dos agentes, possivelmente com diferentes métodos de resolução;

iv) diminuição da interferência entre tarefas, evitando-se interações indesejáveis.

De modo complementar, Klein (1991) define situações cooperativas quando as partes são unidas por um objetivo “super ordenado” para se atingir uma

solução global. Isso implica em sacrifícios das partes menores, onde o objetivo global é dividido e a melhor solução surge da cooperação das partes.

Seguindo a Metodologia do Processo Unificado (Jacobson et al., 1999) e a notação UML de desenvolvimento de software, será apresentado um cenário de Caso de Uso[4] para ilustrar o fluxo de informação e controle no Agente ZEUS genérico (Figura A.6).

1. A Caixa Postal do agente recebe uma mensagem de outro agente.
2. A Caixa Postal passa a mensagem para o Manipulador de Mensagem para ser processada.
3. Ao receber a mensagem, o Manipulador de Mensagem a interpreta como uma solicitação para alcançar uma meta e a repassa para a Máquina de Coordenação.
4. A Máquina de Coordenação determina se a meta deve ser atingida e, se assim for, elabora e coordena um plano de ação apropriado.

Quando a Máquina de Coordenação decide alcançar uma meta, ela invoca o Planejador, que cria um plano para a meta, utilizando as descrições da ação, através de seu Banco de Dados de Planejamento e reservando os recursos que são requeridos pelo plano, disponíveis em seu Banco de Dados de Recursos.

Caso o Planejador não encontre tais recursos em sua base de dados, ele solicita que a Máquina de Coordenação às localize, através de seu Banco de Dados de Conhecimento, quais agentes podem produzir os recursos exigidos.

Não encontrando nenhum agente com as habilidades apropriadas, a Máquina de Coordenação usa a Caixa Postal para enviar uma mensagem a um Facilitador, solicitando uma lista de todos os agentes com as habilidades exigidas. Após a recepção de uma resposta do Facilitador, a Caixa postal remete a

[4] Um Caso de Uso é dito como um conjunto de flames compondo um cenário que descrevem as interações de um ator com o sistema, visando alcançar um objetivo específico. Um ator representa o papel que o agente desempenha no sistema.

mensagem de resposta para a Máquina de Coordenação (através do Manipulador de Mensagem).

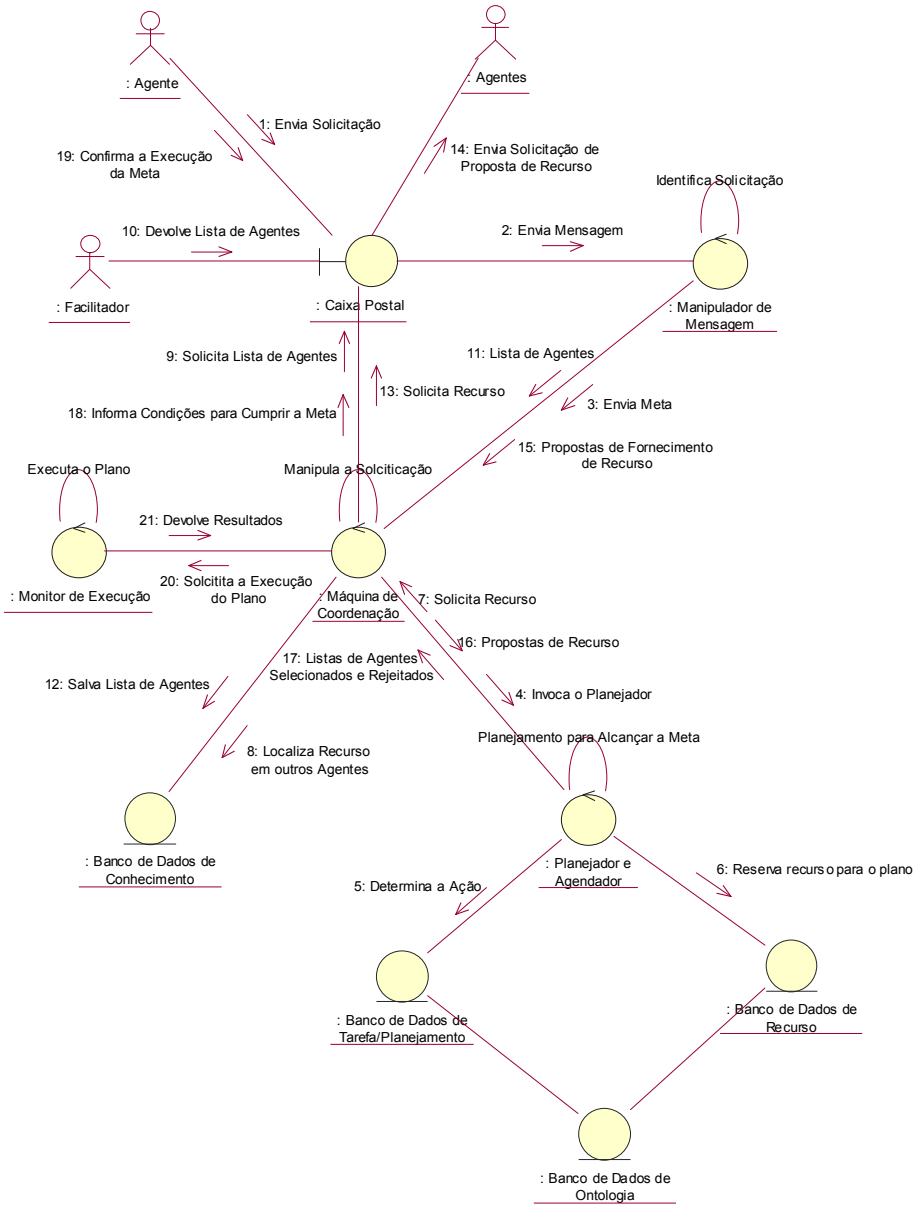


Figura A.6 – Cenário Básico do Agente Zeus (notação UML)

Em seguida, a Máquina de Coordenação salva esta lista de agentes em seu Banco de Dados de Conhecimento e então envia uma mensagem aos agentes pertencentes a esta lista, solicitando-lhes uma proposta de fornecimento do recurso exigido. Novamente as mensagens são enviadas pela Caixa postal e suas respostas devolvidas à Máquina de Coordenação através da Caixa postal e do Manipulador de Mensagem.

Uma vez recebidas as propostas ou expirado o prazo final de resposta dos agentes, a Máquina de Coordenação as envia para o Planejador, que seleciona as mais convenientes para fornecer os recursos exigidos. A conveniência de cada proposta depende de fatores, tais como seu custo e quais se ajustam melhor dentro do plano global para alcançar a meta original. Após a seleção das propostas e o planejamento completo, o Planejador envia duas listas contendo os agentes selecionados e os rejeitados à Máquina de Coordenação, que, por sua vez, enviará aos respectivos agentes uma mensagem de “proposta aceita” ou, se assim for, de “proposta rejeitada”.

Porém, antes de enviar tais mensagens, a Máquina de Coordenação, envia primeiramente uma mensagem ao agente que originou a solicitação para alcançar a meta, informando-o da possibilidade de ser executada e qual o custo envolvido no processo. Só após receber a resposta desse agente, concordando com as condições informadas, é que a Máquina de Coordenação envia as mensagens de proposta aceita ou rejeita aos agentes e informa ao Planejador que o plano para a meta pode ser executado quando conveniente. Caso contrário, após receber uma mensagem desfavorável do agente, a Máquina de Coordenação envia uma mensagem de rejeição para todos os agentes presentes nas listas e ao Planejador é ordenado para abortar o plano.

Uma vez que o plano está pronto para execução, o Monitor de Execução executa as ações especificadas no mesmo, invocando o programa externo declarado em cada descrição da ação. Se o plano inteiro é executado com sucesso, os resultados finais são enviados pela Máquina de Coordenação e Caixa Postal para o agente que solicitou a meta, finalizando o processo.

Como pode ser visto neste cenário, os componentes da Biblioteca de Componente de Agente trabalham em conjunto, fornecendo os agentes disponíveis funcionalmente. A Caixa Postal e o Banco de dados de Ontologia facilitam a comunicação, sendo que o primeiro proporciona aos agentes a habilidade para enviar e receber mensagens em um formato padrão e o segundo permite que cada agente entenda as mensagens recebidas dos outros agentes.

APÊNDICE C - ARQUITETURA TCP/IP

1 CONSIDERAÇÕES INICIAIS

A popularização da Internet fez surgir um novo padrão de transferência de dados, conhecido como a arquitetura TCP/IP. Ela é baseada em quatro camadas, que interagem entre si para fornecer as funcionalidades do modelo de comunicação como um todo.

De fato, a obrigação maior de cada um destes níveis é oferecer determinados serviços para as camadas superiores, escondendo os detalhes de implementação. As camadas são descritas do nível mais baixo até o maior grau de abstração: enlace de dados, rede, transporte e aplicação.

A camada de enlace de dados não é propriamente especificada no modelo de referência TCP/IP. Assim, sua implementação vai depender de vários aspectos técnicos relacionados ao tipo da rede empregada. Entretanto, pode-se dizer que ela é responsável pelo fornecimento de uma interface de serviço à camada de rede sendo implementada através de protocolos de conexão.

A segunda camada, a de rede, tem como função básica garantir que um conjunto de informações (encapsuladas em uma estrutura denominada pacote) seja enviado através da sub-rede de comunicações, da melhor forma possível, em um processo conhecido como roteamento. O nome oficial do protocolo para o envio de dados desta camada é IP (*Internet Protocol*). Existem ainda outros protocolos de controle nesta camada, dentre os quais o ICMP (*Internet Control Message Protocol*).

O terceiro nível, o de transporte, tem por finalidade habilitar a comunicação ponto a ponto entre processos executados em máquinas separadas. Para isso, dois protocolos são especificados: TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*).

A última camada é a de aplicativos. Nela, os usuários podem executar programas que acessam serviços disponíveis através da interligação em redes TCP/IP. Um aplicativo interage com um dos protocolos do nível de transporte para enviar ou receber dados, tais como: os programas de FTP (transferência de arquivos), HTTP (páginas da Internet), SMTP e IMAP (correio eletrônico), DNS (resolvedor de nomes), TELNET (emulador de terminal) e RLOGIN (terminal virtual), SNMP (gerenciamento de rede), NFS (acesso a arquivos) etc.

2 PROTOCOLO IP

O IP (*Internet Protocol*) é protocolo padrão para a comunicação de dados pela Internet. Ele é caracterizado por fornecer um serviço sem conexão e não-confiável, baseado em datagramas. Estes últimos são também conhecidos como pacotes IP e correspondem à unidade básica de transferência deste protocolo.

O serviço é tido como não-confiável porque a entrega não é garantida. Todo o possível é feito para que o pacote seja recebido pelo seu destinatário, entretanto, em alguns casos, como falta de recursos ou falhas na rede, ele pode simplesmente ser perdido, atrasar-se ou ser entregue com problemas, sem que notificações de erro sejam dadas.

No IP, cada pacote é independente um do outro, ou seja, os caminhos por onde os datagramas trafegam são determinados dinamicamente, através dos algoritmos de roteamento deste protocolo. Sendo assim, dois pacotes de uma mesma origem e destinados a um host comum podem seguir rotas distintas.

Para um maior esclarecimento sobre o funcionamento do IP, é preciso conhecer o formato de seu pacote, mais especificamente de seu cabeçalho (Figura C.1).

<table>
<tr><td>VERSÃO</td><td>IHL</td><td>TIPO DE SERVIÇO</td><td colspan="4">COMPRIMENTO TOTAL</td></tr>
<tr><td colspan="3">IDENTIFICAÇÃO</td><td></td><td>D F</td><td>M F</td><td>OFFSET DO FRAGMENTO</td></tr>
<tr><td>TEMPO DE VIDA</td><td colspan="2">PROTOCOLO</td><td colspan="4">CHECKSUM DO CABEÇALHO</td></tr>
<tr><td colspan="7">ENDEREÇO DE ORIGEM</td></tr>
<tr><td colspan="5">ENDEREÇO DE DESTINO</td><td colspan="2">PADDING</td></tr>
<tr><td colspan="7">OPÇÕES (0 OU MAIS)</td></tr>
</table>

Figura C.1: Cabeçalho IP

O cabeçalho do datagrama tem uma parte fixa de vinte bytes e uma parte variável, destinada a possíveis expansões no protocolo. Os campos da parte fixa do cabeçalho mais importantes para este trabalho serão explicados em seguida.

O campo IDENTIFICAÇÃO é um número inteiro de dezesseis bits utilizado para identificar um datagrama. DF significa "DO NOT FRAGMENT" e é uma ordem expressa para que o pacote não seja fragmentado. MF é o acrônimo de "MORE FRAGMENTS" e indica que há mais fragmentos para o pacote. OFFSET DO FRAGMENTO informa a que ponto do datagrama o fragmento pertence.

Os quatro campos acima trabalham em conjunto para controlar a fragmentação e a remontagem do datagrama. A fragmentação acontece devido o pacote, durante o seu percurso, trafegar por diversos tipos de rede, com tamanhos de blocos de transmissão diferentes. Assim, é possível que ele passe por redes onde o MTU (tamanho máximo de unidade) seja tal que não consiga suportar o seu tamanho. Para resolver este problema, o datagrama será fragmentado em pacotes menores e independentes que serão enviados ao destino.

O campo TEMPO DE VIDA especifica o tempo, em segundos, que um datagrama pode continuar existindo. Isto previne situações de “vida eterna” de datagramas devido a erros de rotas ou mesmo *loops.*

O campo PROTOCOLO (8 bits) indica o protocolo usuário do IP, cujos dados são transportados no campo de dados. Pode ser TCP (6), UDP (17) ou ICMP (1).

Os últimos campos relevantes são os ENDEREÇO DE ORIGEM (32 bits) e ENDEREÇO DE DESTINO (32 bits).

3 PROTOCOLO TCP

Este é um dos protocolos da arquitetura TCP/IP que trabalha na camada de transporte, especificado na RFC 793. Seu principal objetivo é fornecer um serviço fim a fim confiável entre processos localizados em *hosts* distintos, independente de quão imperfeitas sejam as redes utilizadas. Para isso, ele busca garantir um fluxo de bytes livre de erros entre as entidades TCP, uma em cada computador.

Para se utilizar este protocolo, é preciso a criação de pontos terminais, conhecidos como *sockets*, entre os pares envolvidos na comunicação. Os *sockets* são formados pela união de um endereço IP e uma porta. Porta é uma abstração usada para definir um determinado serviço dentre os vários que podem rodar em um *host*. Um serviço de FTP, por exemplo, é identificado pela porta 21 e para ser acessado, é preciso que se conheça também o endereço de rede da máquina (onde o serviço está).

A comunicação entre as entidades TCP transmissora e receptora se dá por meio de unidades denominadas segmentos. A Figura C.2 ilustra o formato do cabeçalho de um segmento TCP.

<table>
<tr><td colspan="3">PORTA ORIGEM</td><td>PORTA DESTINO</td></tr>
<tr><td colspan="4">NÚMERO DE SEQUÊNCIA</td></tr>
<tr><td colspan="4">NÚMERO ACK</td></tr>
<tr><td>Hlen</td><td>RESERV</td><td>BITS DE CÓDIGO</td><td>TAMANHO DA JANELA</td></tr>
<tr><td colspan="3">CHECKSUM</td><td>PONTEIRO DE URGÊNCIA</td></tr>
<tr><td colspan="4">OPÇÕES (O OU MAIS)</td></tr>
</table>

Figura C.2: Cabeçalho TCP

O cabeçalho do segmento tem uma parte fixa de vinte bytes e uma parte variável, destinada a possíveis expansões no protocolo. Os campos da parte fixa do cabeçalho relevantes ao presente propósito serão explicados em seguida.

Os campos PORTA ORIGEM e PORTA DESTINO contêm os números da portas TCP associadas às aplicações em cada ponto da conexão.

O campo NÚMERO DE SEQÜÊNCIA identifica o número inteiro seqüencial que identifica um segmento.

O campo NÚMERO ACK *(acknowledgement numbe*r) identifica o número do próximo segmento que a origem espera receber. É interessante observar que o número de seqüência faz o controle do fluxo de dados na transmissão, enquanto o número de ACK faz o controle na direção oposta, ou seja, de recepção.

O campo PONTEIRO DE URGÊNCIA é usado para identificar dados urgentes.

O campo BITS DE CÓDIGO *(code bit*s) contém 6 bits que devem ser usados para determinar o propósito e o conteúdo do segmento. Eles mostram como interpretar os outros campos no cabeçalho. São eles, com suas respectivas funções, caso tenham valor positivo: URG (PONTEIRO DE URGÊNCIA é válido), ACK (NÚMERO ACK é válido), PSH (o segmento deve ser

transmitido logo), RST (reiniciar a conexão), SYN (estabelecer conexões) e FIN (finalizar conexões).

O protocolo TCP é orientado a conexão. Isto implica na necessidade de uma etapa de negociação antes que as informações úteis sejam de fato transmitidas. Este processo se dá em 3 passos, e por isso é conhecido como *Three Way Handshak*e. Eis a explicação: uma máquina A que deseja se comunicar com uma máquina B envia para esta um pacote com o campo SYN ligado. Quando B recebe este pacote, ele envia de volta um datagrama de reconhecimento com os campos SYN e ACK ligados, em resposta ao primeiro pacote. Por fim, A envia para B outro pacote de reconhecimento com o ACK ligado e a negociação se completa.

4 PROTOCOLO UDP

O outro protocolo da camada de transporte da arquitetura TCP/IP é o UDP (*User Datagram Protocol*), especifificado na RFC 768. Ele oferece um serviço não orientado a conexão e não confiável para a comunicação entre dois processos em máquinas distintas. Este protocolo parte do pressuposto que é da aplicação a responsabilidade de fornecer uma transmissão de dados de forma correta. Assim, ele não se preocupa com confirmações de dados recebidos, com o ordenamento das mensagens ou com o controle da velocidade de transmissão entre as máquinas.

O cabeçalho do segmento UDP é retratado na Figura C.3:

PORTA DE ORIGEM	PORTA DE DESTINO
COMPRIMENTO DA MENSAGEM	SOMA DE VERIFICAÇÃO

Figura C.3: Cabeçalho UDP

Os campos relevantes para o entendimento deste trabalho são as PORTA DE ORIGEM e PORTA DE DESTINO.

5 PROTOCOLO ICMP

O ICMP (*Internet Control Message Protocol*) é responsável pelo transporte de mensagens de erro e de controle entre os roteadores ou estações. Ele possibilita a comunicação entre o software IP de uma máquina e o software IP de outra para a transmissão de notificações sobre algum tipo de mau funcionamento de um ponto da rede. Para isso, a mensagem ICMP é encapsulada em um pacote IP e é enviada normalmente.

Este protocolo disponibiliza diversas mensagens de controle. As mais importantes são mostradas na Tabela C.1, com uma breve explicação.

Tabela C.1 - Principais mensagens do protocolo ICMP

Tipo de Mensagem	Explicação
Destinatário inacessível	O pacote não pode ser entregue
Tempo excedido	TEMPO DE VIDA de um pacote expira
Problema de parâmetro	Há erros no cabeçalho do pacote IP
Ajuste de fonte	Pacote que regula a taxa de envio do emissor
Redirecionamento	Mostrar a rota correta de um pacote
Requisição de eco	Verifica se uma máquina está disponível
Resposta ao eco	Confirmar que a máquina está disponível
Marca de tempo	Usado para medir o atraso no envio de pacotes
Resposta a marca do tempo	Usado para medir o atraso no envio de pacotes

O formato de cada mensagem ICMP varia mas existem três campos em comum no cabeçalho:

- Tipo: define o tipo da mensagem (significado e formato), mostrados na Tabela C.1;
- Código: prevê um detalhamento maior sobre o tipo da mensagem;
- Checksum: idêntico ao algoritmo utilizado pelo IP.

	Versão: Draft
Controle de Ações :	Data: 8/out/2000
T1-03	

Controle de Ações

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA toma decisões automáticas ou com auxílio do administrador de segurança, dependendo do nível de alerta do evento gerado pelo agente de avaliação e tendo como base as informações contidas nos bancos de dados de ações (RADB) e estratégia (STDB).

1.1 Contexto

Toma a decisão de ativar outros agentes de captura para coletar mais informações; terminar o processo, usuário ou aplicação; reiniciar a conexão; alterar a permissão de arquivos, usuários ou aplicação; reconfigurar o firewall e notificar o administrador, além de registrar a ocorrência para futuras análises e correções de vulnerabilidades ou com o propósito de um suposto processo judicial.

Ator: Administrador de Segurança e Base de Dados.

1.2 Pré-condições

O agente de controle de ações deve ter recebido as informações do agente de avaliação de segurança e das bases de dados

1.3 Pós-condições

O sistema (ou a rede) não será violado e o administrador de segurança terá um registro da tentativa de intrusão.

2. Fluxo de Eventos

Recebe as informações do agente de avaliação de segurança e age de acordo com as as bases de dados.

2.1 Cenário Básico

1. O agente de avaliação ativa o Agente de Controle de Ações.
2. O agente de controle passa a receber as informações geradas pelo agente de avaliação de segurança e das bases de dados de ações e estratégias.
3. O agente de controle de ações atua no sistema monitorado de acordo com o nível de alerta gerado pelo agente de avaliação de segurança e de acordo com as informações de estratégia e ações contidas nas bases de dados.
4. O agente de controle de ações faz o registro do evento contendo a data, a hora, a origem, o destino, o tipo de evento detectado e que atitude foi tomada pelo sistema.
5. O agente de controle de ações envia ao agente inteligente de segurança local a notificação do evento.
6. O caso de uso termina com sucesso

	Versão: Draft
Controle de Ações :	Data: 8/out/2000
T1-03	

2.2 Cenários Alternativos

2.2.1 Até o passo 2: O agente de controle não sabe que atitude tomar.

3. O agente de controle de ações interage com o agente inteligente de segurança local para solicitar ajuda do administrador na tomada de decisão.
4. O administrador de segurança responde à solicitação ao agente inteligente de segurança local.
5. O agente de controle de ações atua no sistema monitorado de acordo com enviadas pelo agente LSIA.
7. O agente de controle de ações faz o registro do evento contendo a data, a hora, a origem, o destino, o tipo de evento detectado e que atitude foi tomada pelo sistema.
8. O agente de controle de ações envia ao agente inteligente de segurança local a notificação do evento.
9. O caso de uso termina com sucesso

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

4. Requisitos Especiais

5. Questões em Aberto

6. Histórico

Data	Versão	Descrição	Autor
8/out/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
	Draft	Segundo detalhamento	
	1.0	Aprovado	

	Versão: Draft
Atualização :	Data: 8/out/2000
T1-03	

Atualização

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA mantém atualizadas as bases de dados de incidentes de intrusão (DFDB), de padrões de intrusões (IIDB), de ações (RADB) e estratégias (STDB).

1.1 Contexto

Atualiza as bases de dados do sistema e envia status ao agente inteligente de segurança local.

Ator: Administrador de Segurança e Base de Dados.

1.2 Pré-condições

O agente de atualização deve ter recebido as informações do agente inteligente de segurança local e as bases de dados devem estar ativas.

1.3 Pós-condições

As bases de dados ficarão atualizadas e o sistema poderá detectar novos ataques.

2. Fluxo de Eventos

Recebe as informações do agente inteligente de segurança local e as envia para a base de dados especificada.

2.1 Cenário Básico

1. O Administrador de Segurança faz a solicitação de atualizar as bases de dados.
2. O agente inteligente de segurança local pergunta ao administrador quais bases de dados serão atualizadas.
3. O administrador confirma a solicitação, indicando quantas e quais bases de dados serão atualizadas.
4. O agente inteligente de segurança local ativa o agente de atualização para atualizar as bases de dados especificadas.
5. O agente de atualização de segurança atualiza a base de dados de padrões de intrusões (IIDB).
6. O agente de atualização de segurança atualiza a base de dados de incidentes de intrusão (DFDB).
7. O agente de atualização de segurança atualiza a base de dados de ações (RADB).
8. O agente de atualização de segurança atualiza a base de dados de estratégias (STDB).
9. O agente de atualização envia ao agente inteligente de segurança local o status, contendo a data e a hora da última atualização de cada base de dados.
10. O caso de uso termina com sucesso.

O passo 1 será processado caso o Administrador de Segurança queira ativar o agente de atualização manualmente.

No passo 3, o administrador poderá escolher entre 1 e 4 o número de base de dados a serem atualizadas.

Os passos 5, 6, 7 e 8 serão processados de acordo com que for especificado no passo 3.

	Versão: Draft
Atualização :	Data: 8/out/2000
T1-03	

2.2 Cenários Alternativos

2.2.1 Antes do passo 4: O agente de controle de ações deseja atualizar as bases de dados automaticamente.

1. O agente de controle de ações faz a solicitação de atualizar as bases de dados.
2. O agente inteligente de segurança local pergunta ao agente de controle de ações quais bases de dados serão atualizadas.
3. O agente de controle de ações confirma a solicitação, indicando quantas e quais bases de dados serão atualizadas.

Os passos seguintes serão processados da mesma forma que o cenário principal.

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

4. Requisitos Especiais

5. Questões em Aberto

6. Histórico

Data	Versão	Descrição	Autor
8/outubro/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
	Draft	Segundo detalhamento	
	1.0	Aprovado	

	Versão: Draft
Avaliação de Segurança :	Data: 28/outubro/2000
T1-03	

Avaliação de Segurança

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA verifica a autenticidade dos resultados obtidos pelo agente de monitoramento baseados em um conjunto de dados disponíveis nas bases de dados de incidentes de intrusão (DFDB) e estratégias (STBD) e envia um nível de alerta para o agente controlador.

1.1 Contexto

Identificar eventos suspeitos e enviar um nível de alerta ao agente controlador.

Ator: Administrador de Segurança.

1.2 Pré-condições

O SEA deve ter recebido as informações do agente de monitoramento e das bases dados de incidentes de intrusão (DFDB) e estratégias (STBD).

1.3 Pós-condições

O agente envia um nível de alerta para o agente controlador.

2. Fluxo de Eventos

Recebe as informações do agente de monitoramento e da base de incidentes de intrusão e estratégias, identifica a intrusão ou tentativas de intrusão e envia um alerta para o agente controlador de ações.

2.1 Cenário Básico

1. O agente de monitoramento ativa o agente de avaliação de segurança.
2. O agente de avaliação de segurança passa a receber as informações geradas pelo agente de monitoramento, em tempo real ou em intervalos de tempo.
3. O agente de avaliação verifica a autenticidade das informações, de acordo com as bases de dados de incidentes de intrusão e de estratégias.
4. O agente de avaliação de segurança envia ao agente controlador um nível de alerta associado ao grau de risco representado pelo evento detectado.
5. O caso de uso termina com sucesso

2.2 Cenários Alternativos

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

4. Requisitos Especiais

5. Questões em Aberto

	Versão: Draft
Avaliação de Segurança :	Data: 28/outubro/2000
T1-03	

6. Histórico

Data	Versão	Descrição	Autor
28/10/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
	Draft	Segundo detalhamento	
	1.0	Aprovado	

	Versão: Draft
Integridade de Agente :	Data: 8/out/2000
T1-03	

Integridade de Agente

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA garante a sua integridade. O agente de integridade busca por eventos não esperados ou diferentes do perfil normal dos agentes ativos do sistema. Caso ocorra alguma anomalia, este envia informações quanto a situação de cada agente do sistema, atribuindo um nível de alerta (normal ou anormal) para o agente inteligente de segurança local, que por sua vez, notificará o administrador.

1.1 Contexto

Detecta anormalidade dos agentes e envia um nível de alerta ao administrador.

Ator:

1.2 Pré-condições

O agente de integridade deve ter um perfil de cada agente.

1.3 Pós-condições

O administrador terá condições de verificar possíveis ataques ao próprio sistema de detecção de intrusão.

2. Fluxo de Eventos

Coleta informações dos agentes ativos do sistema NIDIA e compara com seus perfis esperados e envia ao agente inteligente de segurança local o resultado da análise.

2.1 Cenário Básico

1. O agente de integridade do sistema coleta informações dos agentes do sistema NIDIA.
2. Após um determinado tempo, traça um perfil para cada agente supervisionado.
3. O agente de integridade compara a atividade dos agentes com perfis de comportamento esperados para cada um.
4. Caso seja detectado alguma anomalia, o agente controlador de ações é ativado.
5. O resultado é enviado para o agente inteligente de segurança local.
6. O agente inteligente de segurança local notifica o administrador.
7. O caso de uso termina com sucesso.

Quando o agente controlador de ações é ativado (passo 4), os passos subseqüentes são descritos no cenário básico de controle de ações. Quando finalizados, retorna-se ao passo 5.

2.2 Cenários Alternativos

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

	Versão: Draft
Integridade de Agente :	Data: 8/out/2000
T1-03	

4. Requisitos Especiais

5. Questões em Aberto

6. Histórico

Data	Versão	Descrição	Autor
8/outubro/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
	Draft	Segundo detalhamento	
	1.0	Aprovado	

	Versão: 1.0
Monitoramento :	Data: 05/outubro/2000
T1-03	

Monitoramento

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA identifica padrões de ataque, através de uma rede neural, comparando-as com a base de dados de padrões de intrusão (IIDB) dos eventos coletados pelos agentes sensores e formatando os eventos considerados suspeitos e enviando-os para serem analisados pelo agente de avaliação de segurança.

1.1 Contexto

Identificar e formatar eventos suspeitos e enviar para o agente de avaliação de segurança.

Ator: Administrador de Segurança e Base de Dados.

1.2 Pré-condições

O agente de monitoramento deve ter recebido as informações dos agentes sensores e da base de dados de padrões de intrusão (IIDB).

1.3 Pós-condições

O agente envia os dados para serem analisados pelo agente de avaliação de segurança.

2. Fluxo de Eventos

Recebe as informações dos agentes sensores, identifica, formata os eventos considerados suspeitos e os envia para o agente de avaliação de segurança.

2.1 Cenário Básico

1. O agente sensor ativa o agente de monitoramento.
2. O agente de monitoramento passa a receber as informações geradas pelos agentes sensores em tempo real ou em intervalos de tempo.
3. O agente de monitoramento compara as informações recebidas com a base de dados de padrões de intrusão.
4. O agente de monitoramento organiza os eventos suspeitos numa relação de causa e efeito.
5. O agente de monitoramento formata os dados, associando-os a um grau de risco, para serem enviados ao agente de avaliação de segurança.
6. O agente monitoramento envia as informações para o agente de avaliação de segurança.
7. O caso de uso termina com sucesso

2.2 Cenários Alternativos

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

	Versão: 1.0
Monitoramento :	Data: 05/outubro/2000
T1-03	

4. Requisitos Especiais

5. Questões em Aberto

6. Histórico

Data	Versão	Descrição	Autor
0510/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
	Draft	Segundo detalhamento	
	1.0	Aprovado	

	Versão: 1.0
Sensor de Host :	Data: 05/outubro/2000
T1-03	

Sensor de Host

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA captura informações de servidores críticos para serem avaliadas com o objetivo de detectar atividades anormais.

1.1 Contexto

Capturar informações do servidor e enviar para o agente de monitoramento.

Ator: Administrador de Segurança.

1.2 Pré-condições

O agente sensor de host, o agente de monitoramento e o servidor devem estar ativos e o sistema deve estar auditando os eventos desejados.

1.3 Pós-condições

O agente envia as informações coletadas para o agente de monitoramento.

2. Fluxo de Eventos

Lê o arquivo de log do servidor e envia para o agente de monitoramento.

2.1 Cenário Básico

1. O Administrador de Segurança indica que deseja ativar o agente sensor de host.
2. O Administrador de Segurança identifica o Agente Sensor de Host a ser ativado, através do Agente Inteligente de Segurança Local (LSIA) e solicita a sua ativação.
3. O administrador de Segurança confirma a solicitação de ativação do agente sensor de host.
4. O agente LSIA ativa o agente selecionado e mostra o novo *status* do agente sensor de host para o Administrador de Segurança.
5. O agente sensor de host passa a coletar informações em tempo real, ou em intervalos de tempo, dos arquivos de log do servidor que está sendo monitorado.
6. O agente sensor de host envia as informações coletadas para o agente de monitoramento.
7. O caso de uso termina com sucesso

Os passos 1 até 4 serão processados e/ou repetidos caso o Administrador queira ativar um ou mais agentes sensores de host.

2.2 Cenários Alternativos

2.2.1 Até o passo 2: O Administrador de Segurança deseja cancelar a ativação do agente

3. O administrador cancela a confirmação de ativação do agente sensor de host.
4. O caso de uso termina com sucesso.

	Versão: 1.0
Sensor de Host :	Data: 05/outubro/2000
T1-03	

2.2.2 O agente LSIA deseja ativar o agente sensor de host sem intervenção do administrador.

1. O agente LSIA ativa o agente sensor de host e mostra o *status* para o Administrador de Segurança.
8. O agente sensor de host passa a coletar informações em tempo real, ou em intervalos de tempo, dos arquivos de log do servidor que está sendo monitorado.
9. O agente sensor de host envia as informações coletadas para o agente de monitoramento.
2. O caso de uso termina com sucesso

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

4. Requisitos Especiais

5. Questões em Aberto

6. Histórico

Data	Versão	Descrição	Autor
05/10/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
07/11/00	Draft	Segundo detalhamento	Christiane F. L. Lima
12/12/00	1.0	Aprovado	Christiane F. L. Lima

	Versão: 1.0
Sensor de Rede :	Data: 05/outubro/2000
T1-03	

Sensor de Rede

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA captura informações da rede para serem avaliadas com o objetivo de detectar conexões suspeitas e comportamentos anormais.

1.1 Contexto

Capturar informações da rede e enviar para o agente de monitoramento.

Ator: Administrador de Segurança.

1.2 Pré-condições

O agente sensor de rede e o agente de monitoramento devem estar ativos e o segmento de rede a ser monitorado deve existir.

1.3 Pós-condições

O agente envia as informações coletadas para o agente de monitoramento.

2. Fluxo de Eventos

Captura os pacotes da rede e envia para o agente de monitoramento.

2.1 Cenário Básico

1. O Administrador de Segurança indica que deseja ativar o agente sensor de rede.
2. O Administrador de Segurança identifica o Agente Sensor de Rede a ser ativado, através do Agente Inteligente de Segurança Local (LSIA) e solicita a sua ativação.
3. O administrador de Segurança confirma a solicitação de ativação do agente sensor de rede.
4. O agente LSIA ativa o agente selecionado e mostra o novo *status* do agente sensor de rede para o Administrador de Segurança.
5. O agente sensor de rede passa a coletar informações em tempo real, ou em intervalos de tempo, dos pacotes que estão trafegando na rede.
6. O agente sensor de rede envia as informações coletadas para o agente de monitoramento.
7. O caso de uso termina com sucesso

Os passos 1 até 4 serão processados e/ou repetidos caso o Administrador queira ativar um ou mais agentes sensores de rede.

2.2 Cenários Alternativos

2.2.1 Até o passo 2: O Administrador de Segurança deseja cancelar a ativação do agente

3. O administrador cancela a confirmação de ativação do agente sensor de rede.
4. O caso de uso termina com sucesso.

	Versão: 1.0
Sensor de Rede :	Data: 05/outubro/2000
T1-03	

2.2.2 O agente LSIA deseja ativar o agente sensor de rede sem intervenção do administrador.

1. O agente LSIA ativa o agente sensor de rede e mostra o *status* para o Administrador de Segurança.
2. O agente sensor de rede passa a coletar informações em tempo real, ou em intervalos de tempo, dos pacotes que estão trafegando na rede.
3. O agente sensor de rede envia as informações coletadas para o agente de monitoramento.
4. O caso de uso termina com sucesso

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

4. Requisitos Especiais

5. Questões em Aberto

6. Histórico

Data	Versão	Descrição	Autor
05/10/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
10/11/00	Draft	Segundo detalhamento	Christiane F. L. Lima
07/12/00	1.0	Aprovado	Christiane F. L. Lima

	Versão: Draft
Segurança Local :	Data: 8/out/2000
T1-03	

Segurança Local

1. Informações Características

Ao utilizar este caso de uso o sistema NIDIA coordena a sociedade de agentes e interage com o administrador de segurança para controlar o sistema, bem como ter uma visão geral do ambiente monitorado.

1.1 Contexto

Coordena a sociedade de agentes e interage com o administrador de segurança e as bases de dados.
Ator: Administrador de Segurança e as Base de Dados.

1.2 Pré-condições

O agente inteligente de segurança local deve estar se comunicando com todos os agentes da sociedade.

1.3 Pós-condições

O administrador terá condições de interagir com o sistema e monitorar o ambiente computacional.

2. Fluxo de Eventos

Troca informações com os agentes ativos do sistema NIDIA, acessa as bases de dados, interage com o administrador.

2.1 Cenário Básico

1. O agente inteligente de segurança local do sistema coleta informações dos agentes do sistema NIDIA.
2. Interage com as bases de dados.
3. Interage com o administrador.
4. O agente inteligente de segurança local faz notificações ao administrador.
5. O caso de uso termina com sucesso.

2.2 Cenários Alternativos

2.3 Pontos de Extensão

Nada por enquanto.

2.3.1 <name of extension point>

3. Variações de Dados e de Tecnologia

Nada por enquanto.

4. Requisitos Especiais

5. Questões em Aberto

	Versão: Draft
Segurança Local :	Data: 8/out/2000
T1-03	

6. Histórico

Data	Versão	Descrição	Autor
8/outubro/00	Draft	Primeiro detalhamento	Christiane F. L. Lima
	Draft	Segundo detalhamento	
	1.0	Aprovado	