Agentes Móveis e Sistemas de Gerenciamento SNMP

Antonio José dos Santos Brandão, Edson dos Santos Moreira
{abrandao, edson}@icmc.sc.usp.br

Mathematics and Computing Institute, Universidade de São Paulo
Cx. Postal 668 - CEP 13560-970
São Carlos - São Paulo - Brasil

Resumo
Após a grande popularização da Internet, segurança passou a ser palavra de ordem. Este trabalho apresenta a integração de dois projetos. O primeiro, denominado *NetTracke*r, é um sistema de gerenciamento que utiliza as facilidades do protocolo SNMP. O segundo, constitui-se de um Sistema de Detecção de Intrusão baseado em agentes móveis.
Para esta integração, foi desenvolvido um agente móvel que funciona como elo de ligação entre os dois, recebendo mensagens do Sistema de Detecção de Intrusão através de comunicação entre agentes e repassando-as ao *NetTracker* através do SNMP. Esta união habilita ao Sistema de Detecção de Intrusão utilizar recursos do *NetTracker* como contramedidas a intrusos detectados.

Palavras chave: Gestão Integrada de Segurança, Gerenciamento de Redes, Agentes Móveis.

Abstract
Security started to be a watchword due to the ever increasing worldwide use of the Internet. This work presents the integration of two systems. The first one, called NetTracker, is a network management system that uses the easiness of SNMP protocol. The second one consists of an Intrusion Detection System based on mobile agents.
To fulfil this integration, a mobile agent has been developed. It works as a link between the two systems, receiving messages from the Intrusion Detection System and forwarding them to NetTracker. This union qualifies the Intrusion Detection System to use NetTracker's features to counter to the detected intruders.

Keywords: Integrated Security Management, Network Management, Mobile Agents.

Introdução

A popularização da Internet já é uma realidade e uso dela para os mais diversos fins implica na pesquisa de tecnologias de segurança que acompanhem esse crescimento. Nos últimos meses o volume de notícias de invasões tem aumentado bastante. Segundo previsões do Instituto IDC (Internacional Data Corporation) o mercado de segurança em informática movimentará US$ 11 bilhões no ano de 2004.

Em [MOURO, 1997] e [MOURO; MORISHITA & MOREIRA, 1997] é apresentado um sistema de gerenciamento de redes chamado *NetTracke*r. Esse sistema tem como característica principal o uso de recursos da WWW para seu acesso o que torna possível ativar módulos de gerenciamento remotamente e observar os resultados de sua execução através de páginas HTML. A estrutura desse gerenciador é a mais modular possível, de forma que partes possam ser inseridas ou removidas sem a necessidade de grandes alterações no sistema.

Atualmente, os softwares denominados agentes móveis e seu uso no gerenciamento de segurança têm sido alvo de grande atenção nas pesquisas [REAMI, 1998] [BERNARDES, 2000]. O uso dos agentes móveis acrescenta características importantes para Sistemas de Detecção de Intrusão porém pouco encontradas nos sistemas atuais. Estas características incluem bom desempenho, escalabilidade, fácil configuração, tolerância à falhas e execução contínua.

O objetivo deste trabalho foi o desenvolvimento de um agente que possa fazer a integração do Sistema de Detecção de Intrusão baseado em agentes móveis apresentado em [BERNARDES, 2000] ao ambiente *NetTracker* [MOURO, 1997], provendo formas de gerenciamento de segurança através da utilização das facilidades do protocolo SNMP.

Um ponto importante na leitura deste projeto é a atenção em relação às denominações utilizadas. Quando estudamos o protocolo SNMP a palavra agente tem como significado o processo que é executado por um recurso gerenciado. Ao estudar os Agentes Móveis o significado é diferente: programas que entre outras características possuem a capacidade de mover-se na rede e se executarem de modo assíncrono em várias máquinas.

Tratando-se de SNMP objetos são apenas variáveis que guardam informações sobre o estado dos dispositivos gerenciados. Quando Agentes Móveis são estudados o conceito de objetos se encaixa dentro do usado no paradigma de programação orientada a objetos — o que é bastante diferente dos objetos SNMP.

A seguir, a sessão 2 é destinada à revisão do protocolo SNMP e suas vantagens, a sessão 3 trata dos Sistemas de Detecção de Intrusão, a sessão 4 guarda considerações sobre a linguagem Java e os Agentes Móveis, em 5 é descrito o *framework Aglet* de desenvolvimento de agentes móveis, em 6 é apresentado o agente implementado e, finalmente, na sessão 7 são feitas as considerações finais.

Protocolo SNMP

Com a expansão das redes de computadores o protocolo SNMP foi designado para suprir a necessidade de uma ferramenta de gerenciamento estruturado e sistemático. A primeira versão do protocolo foi definida no RFC 1155. Este protocolo foi amplamente implementado em produtos comerciais e logo foi aceito como padrão para gerenciamento de redes. Mais adiante, nos RFCs 1441 a 1452 uma nova versão do protocolo SNMP foi desenvolvida, o SNMPv2, que veio devido à deficiência de segurança do SNMPv1 e sua ineficiência quando utilizado em ambientes grandes e complexos.

Recentemente, em 1999, mais uma versão do protocolo foi especificada: o SNMPv3, encontrado no RFC 2570. Vale lembrar que SNMPv3 não é um novo padrão que substitui as versões anteriores. O RFC 2271, que é um dos documentos que o definem, descreve uma arquitetura em que todas as versões atuais (e possivelmente as futuras) do SNMP se encaixam.

A idéia básica de todo mecanismo de gerenciamento é que dispositivos podem ser configurados de duas formas: ou para serem gerenciados ou para serem os gerentes. Seguindo essa idéia o modelo de gerenciamento do SNMP consiste em:

- Diversos nós gerenciados, cada um com uma entidade SNMP que provê acesso remoto às informações gerenciadas — geralmente chamados de agentes.
- Pelo menos uma entidade que rode aplicações de gerenciamento — chamada de gerente.
- Um protocolo de gerenciamento usado para trocar as informações entre gerente e agentes (o protocolo SNMP).
- As informações gerenciadas.

Os nós gerenciados podem ser *hosts*, roteadores, pontes, impressoras ou qualquer outro dispositivo capaz de trocar informações com os gerentes. Os agentes presentes nos

dispositivos gerenciados são responsáveis por coletar e manter informações sobre o ambiente local, oferecendo essas informações ao gerente quando solicitadas ou quando algum evento importante ocorrer e aceitar os comandos enviados pelo gerente para que a configuração local seja alterada.

Os agentes são desenvolvidos de forma a serem o mais simples possível deixando toda a tarefa de processamento para as estações de gerenciamento.

Estações de gerenciamento geralmente provêem interfaces para que o gerente da rede possa observá-la e controlá-la. Por exemplo, observar o uso da largura de banda, descobrir *dead-links*, alterar tabelas de roteamento, etc. Processos são executados na estação de gerenciamento para que se comuniquem com os agentes através da rede e obtenham essas informações. Essa comunicação é feita através de comandos *get* e *set* e recebimentos de avisos *trap*. [STALLINGS, 1998].

A partir do SNMPv2 os gerentes também podem interagir com outros gerentes melhorando assim a performance em ambientes de gerenciamento descentralizado. Cada dispositivo mantém uma ou mais variáveis (objetos) que descrevem seu estado. A coleção de todos os objetos de uma rede constitui uma MIB (*Management Information Base*).

O software implementado neste projeto define uma MIB contendo as informações passadas pelo sistema de detecção de intrusão e que é requisitada pelo *NetTracker*.

Sistemas de Detecção de Intrusão

Com a crescente preocupação na área de segurança cresce também a tentativa de desenvolvido de Sistemas de Detecção de Intrusão (SDIs) que protejam a rede respondendo aos crescentes e sofisticados ataques. A literatura apresenta uma vasta quantidade de propostas para Sistemas de Detecção de Intrusão. Entre elas, destacam-se: o IDES [LUNT & JAGANNATHAN, 1998]; o NIDES [JAVITZ et al., 1993]; o EMERALD [PORRAS & NEUMANN, 1997]; o sistema de detecção baseado em Data *Minning* apresentado em [LEE & STOLFO, 1997]; o ACME! [CANSIAN et al., 1997] [BONIFÁCIO Jr et al., 1998a] e o sistema apresentado em [BERNARDES, 2000].

A detecção de intrusão pode ser definida como o problema de identificar indivíduos que utilizam ou tentam utilizar o sistema computacional sem autorização (*crackers*) e aqueles que possuem acesso legítimo ao sistema mas que abusam ou tentam abusar dos seus privilégios (ataques internos). Sistemas que têm como objetivo a solução deste problema são chamados Sistemas de Detecção de Intrusão.

Os Sistemas de Detecção de Intrusão geralmente são divididos em duas categorias: baseados em anomalia e em abuso.

No modelo baseado em anomalia, a intrusão é detectada pelo uso fora do padrão do sistema. Estatísticas de uso são confeccionadas e o SDI observa qualquer utilização que fuja significantemente destas estatísticas. Como exemplo, o uso noturno da conta de um usuário que trabalha durante o dia pode significar uma intrusão.

No modelo baseado em abuso o sistema tem conhecimento prévio de padrões de ataques e procura por qualquer utilização do sistema que contenha alguma assinatura de ataque como uma certa seqüência de operações ou o uso de certos arquivos do sistema.

Bernardes apresentou um sistema híbrido, possuindo características de sistemas baseados em anomalia e de sistemas baseados em abuso [BERNARDES, 2000].

Abaixo, seguem algumas características que são esperadas de Sistemas de Detecção de Intrusão. Tais características estão presentes no sistema proposto por Bernardes:

- Ser executado continuamente, com a mínima interferência humana,
- Ser tolerante à falhas, ou seja, capaz de se recuperar após falhas do sistema, sejam elas acidentais ou causadas por invasores.

- Ser capaz de se automonitorar e detectar se foi modificado por um invasor.
- Ser flexível, capaz de ser configurado de acordo com a política de segurança do sistema onde será utilizado. Também deve ser capaz de se adaptar às mudanças no sistema, quando novos usuários, aplicações ou máquinas são adicionados.
- Ser capaz de monitorar uma grande quantidade de sites provendo resultados de maneira precisa.
- Ser modular e implementado de forma que quando um módulo pare de funcionar altere o mínimo possível o funcionamento do resto do sistema.
- Permitir configuração dinâmica. Quando um grande número de sites está sob monitoramento é impraticável reinicializar o SDI cada vez que alguma configuração é alterada.

Agentes Móveis e Linguagem Java

A linguagem Java, desenvolvida pela Sun Microsystems, apresenta portabilidade, clareza, fácil aprendizado e orientação a objeto. O compilador Java traduz programas fonte Java em um código intermediário e independente da plataforma chamado *Java Byte Code*, que é interpretado por uma *Java Virtual Machine* (JVM).

Por estes motivos a linguagem Java é a mais utilizada quando se visa a portabilidade do código. É o caso dos agentes móveis. Além da vantagem da portabilidade, a linguagem Java apresenta outras características que a fazem uma boa linguagem para programação de agentes móveis [LANGE & OSHIMA, 1998]:

- Execução segura: A linguagem é do tipo *tape safe*, isto é, não permite a atribuição de valores de tipos diferentes do especificado para cada variável. Não permite também sobrescrita em memória. Isto impossibilita qualquer ataque do tipo *buffer overflow* e torna relativamente segura a execução agentes não confiáveis.
- Carregamento dinâmico de classes: Permite carregar classes em tempo de execução. Isto torna possível a execução de agentes cada um em seu espaço, fazendo-os mais independentes e seguros.
- Programação *Multithread*: Permite que cada agente seja executado em *thread* de execução (ou *lightweight process*) contribuindo para o comportamento autônomo entre agentes.
- Reflexão: Códigos em Java podem descobrir informações sobre campos, métodos e construtores de classes carregadas, podendo utilizá-los para operar os objetos. Esta característica acomoda a necessidade dos agentes agirem inteligentemente sobre si mesmos e sobre outros agentes.

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

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

Franklin e Graesser utilizam a seguinte definição: “*Um agente autônomo é um sistema situado em um ambiente, e parte do mesmo, percebendo este ambiente e agindo sobre*

o mesmo no decorrer do tempo, de acordo com sua própria agenda e de modo a afetar o que ele perceberá no futuro" [FRANKLIN & GRAESSER, 1996] *apud* [BERNARDES, 2000].

O Modelo *Aglet*

O termo *Aglet* surge da junção das palavras *agent* e *applet*. O modelo foi desenhado para se beneficiar das características dos agentes em Java.

Existem oito operações básicas que podem ser realizadas com o *Aglet*: *creation*, *cloning, dispatching, retraction, deactivation, activation, disposal* e *messaging.*

- *Creation*. Um novo *Aglet* recebe um identificador, é inserido em um contexto e é inicializado. Sua execução começa logo em seguida.
- *Clonning*. Trata-se de uma clonagem quase perfeita do *Aglet*. O *Aglet* resultante difere do original apenas pelo seu identificador e pelo fato da execução do novo *Aglet* partir do início. Os *threads* de execução não são clonados.
- *Dispatching*. Remove um *Aglet* de um contexto e o insere em outro, onde sua execução será reiniciada. Costuma-se dizer que o *Aglet* foi "empurrado" para seu novo contexto.
- *Retraction*. Retira o *Aglet* de seu contexto atual e o insere no contexto de onde foi solicitada a operação. É o inverso de *Dispatching*.
- *Activation* e *deactivation*. A operação de *desactivation* temporariamente bloqueia a execução do *Aglet* e armazena seu estado de execução em disco. *Activation* é a operação que restaura o *Aglet* ao seu contexto.
- *Disposal*. Pára a execução do *Aglet* e o remove de seu contexto atual.
- *Messaging*. Envio, recebimento e o manejamento de mensagens entre *Aglets*.

A Figura 1, presente em [LANGE & OSHIMA, 1998] (artigo dos autores da API *Aglet*), ilustra as operações acima apresentando o modelo do ciclo de vida de um *Aglet*.

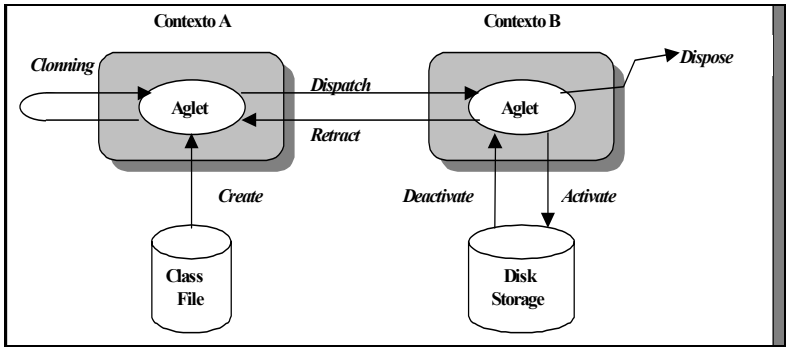

Figura 1 - Modelo do Ciclo de vida de um *Aglet*

Aglet é o kit que o programador utiliza para criar e operar agentes. Esta API apresenta um conjunto de classes e interfaces para sua manipulação: *Aglet*, *AgletProxy*, *AgletContext* e *Message* e também métodos que efetuam as operações básicas sobre o *Aglet* (*creation, cloning, dispatching, retraction, deactivation, activation, disposal* e *messaging*), facilitando muito a tarefa de criar e manipular agentes.

Neste projeto optou-se pelo uso do *framework Aglet* visando manter a padronização e facilitar a comunicação com o SDI proposto por Bernardes [BERNARDES, 1999].

Agente de Integração

Para a implementação do Agente de Integração a API *JMGMT (Free Java SNMP Stack with Servlet and Agent Framework*) foi utilizada. Ela provê uma implementação do

SNMP criando uma abstração das particularidades deste protocolo de gerenciamento facilitando assim o desenvolvimento de aplicações que o utilizam.

Durante as pesquisas diversas *APIs* JAVA que trabalham com SNMP foram encontradas. A escolha por esta foi feita por tratar-se de um pacote pequeno, com código fonte disponível e por ser aconselhada pelo time de desenvolvimento da API *Aglet*. Seu código fonte e documentação podem ser encontrados em http://i31www.ira.uka.de/~sd/manager/jmgmt/.

O Agente de Integração funciona basicamente pelo do recebimento e resposta de mensagens SNMP e do recebimento de mensagens de outros agentes móveis. A Figura 2 mostra a arquitetura formada com a implementação do Agente de Integração:

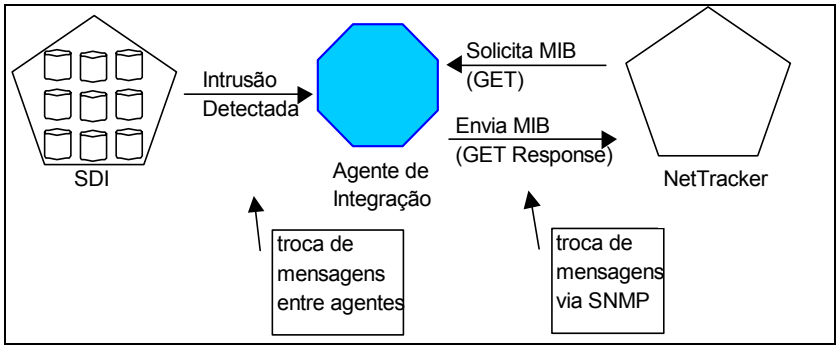

Figura 2 – Arquitetura resultante da implementação do agente.

Quando o Agente de Integração recebe uma mensagem do SDI, prepara a MIB e passa a aguardar requisições feitas pelo gerenciador *NetTracker*. A API *Aglet* define um método *handleMessage* que é disparado sempre que uma mensagem chega para o Agente. Assim, neste método foram colocados os procedimentos para definição da MIB. Outro método, *run,* é executado sempre que o agente é ativado. Ali são colocados procedimentos de inicialização de variáveis e das portas de comunicação SNMP. Finalmente, o método *loop* é executado continuamente para atender requisições SNMP, chamar métodos para decodificá-las, gerar a resposta e envia-la à origem da requisição. A Figura 3 mostra a interação entre os métodos principais: *handleMessage* e *loop.*

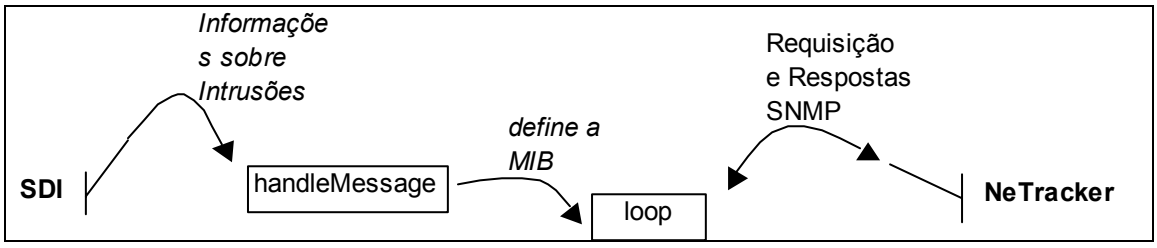

Figura 3 – Interação entre os principais métodos do agente.

É importante ressaltar que apesar do sistema completo do SDI ser composto por diversos agentes móveis, neste trabalho foi implementado um agente estacionário que serviu de integração entre o SDI e o Sistema de Gerenciamento *NetTracker*.

O código fonte do agente implementado neste trabalho pode ser encontrado em http://www.grad.icmc.sc.usp.br/~abrandao/ic/fapesp/.

Considerações Finais

No início do desenvolvimento desse trabalho, a implementação de um Agente Móvel era tarefa vista como complicada. Porém, mostrou-se de forma diferente. As facilidades da API *Aglet* com suas características orientadas a eventos permitem que em poucas linhas de código se transforme determinado aplicativo em um Agente por meio da herança de classes e da implementação de alguns métodos.

Apesar da API *Aglet* nos ter ajudado neste ponto, muito tempo foi perdido em detalhes da utilização da mesma já que não é compatível com as versões mais recentes da linguagem Java. Somente nas últimas semanas foi disponibilizada uma versão compatível com a plataforma Java 2.

A manipulação de informações trocadas no protocolo SNMP também se mostrou mais fácil que o esperado devido à abstração criada pela API JMGMT. Caso nenhuma ferramenta deste tipo fosse utilizada o projeto se tornaria muito mais trabalhoso.

Referências Bibliográficas

[BERNARDES, 1999] BERNARDES, M.C. Avaliação do Uso de Agentes Móveis em Segurança Computacional. Dissertação de Mestrado, 112 p. ICMC/USP - São Carlos, 1999.

[BERNARDES, 2000] BERNARDES, M. C.; MOREIRA, E.S.; An Architecture for an Intrusion Detection System Based on Mobile Agents. In: *International Symposium on Advanced Distributed Systems*, Guadalajara, Mexico, Mar/2000

[BONIFÁCIO Jr. et al. 1998a] BONIFÁCIO Jr., J. M.; CANSIAN, A.M.; CARVALHO, A.C.P.L; MOREIRA, E.S. Neural Networks Applied in Intrusion Detection Systems. In: *Proceedings of the IEEE IJCNN '98 International Joint Conference on Neural Networks*, Anchorage, Alaska, Maio 1998.

[CANSIAN et al. 1997] 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.

[ENDLER, 1998] ENDLER, Markus. "Novos Paradigmas de Interação usando Agentes Móveis" Departamento de Ciência da Computação. IME. USP. SBRC98.

[FRANKLIN & GRAESSER, 1996] FRANKLIN, S.; GRAESSER, A. Is It An Agent, or Just a Program? A Taxonomy for Autonomous Agents. In: *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag. 1996. Disponível on-line em: http://www.msci.memphis.edu/~franklin/AgentProg.html Visitado em 16/04/2002

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

[JAVITZ et al, 1993] JAVITZ, H.S.; VALDES, A.; LUNT, T.F.; TAMARU, A.; TYSON, M.; LOWRANCE, J. Next Generation Intrusion Detection Expert System ({NIDES}). 1993. (Relatório Técnico SRI-A016-Rationales).

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

[LEE & STOLFO 1997] LEE, W.; STOLFO, S. Data Mining Approaches for Intrusion Detection. In: *Proceedings 1998 7th USENIX Security Symposium*. 1997.

[LUNT & JAGANNATAHAN, 1988] LUNT, T.F.; JAGANNATHAN, R. A Prototype Real-Time Intrusion Detection Expert System. In: *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, Abril 1988.

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

[MOURO, 1997] MOURO, R. B. Uma Arquitetura Operacional Extensível para Ferramentas de Gerenciamento de Redes. São Carlos, 1997. 60p. Dissertação (Mestrado) - Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.

[MOURO, MORISHITA & MOREIRA 1997] MOURO, R. B.; MORISHITA, F. T.; MOREIRA, E. S. NetTracker: Uma Arquitetura Operacional Extensível Para Ferramentas de Gerenciamento de Redes. São Carlos, 1997. *In: Proceedings of the XV Brazilian Symposium on Computer Networks*, SBRC'97, p. 164-174, São Carlos, Maio 1997.

[PORRAS & NEUMANN, 1997] PORRAS, P.A.; NEUMANN, P.G. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In: *Proceedings of 20th National Information Systems Security Conference Proceedings*, p.353-366. Baltimore, Maryland. Outubro 1997. Disponível on-line em: http://www.csl.sri.com/emerald/emerald-niss97.html Visitado em 16/04/2002.

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

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

[STALLINGS, 1998] Stallings W., *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Third Edition, 1999. ISBN: 0-201-48534-6

[TANENBAUM, 96] Tanenbaum, Andrew S., *Computer Networks*. Third Edition. 1996. ISBN: 0-13-349945-6