MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-10207-PUD/133

SALL: SISTEMA DE APOIO À ANÁLISE DE LOGS

Lília de Sá Silva

INPE
São José dos Campos
2003

SAAL: Sistema de Apoio à Análise de *Logs*

Daniel Merli Lamosa
Instituto Nacional de Pesquisas Espaciais
Pós-graduação em Computação Aplicada
lamosa@lac.inpe.br

Lília de Sá Silva
Instituto Nacional de Pesquisas Espaciais
Pós-graduação em Computação Aplicada
lilia@dss.inpe.br

Resumo

Este trabalho apresenta um sistema baseado em agentes que utiliza o conceito de 'Artificial Ignorance' para realizar a filtragem de entradas de logs de servidores Web. A arquitetura geral do sistema implementado e a análise dos resultados são assuntos discutidos, bem como são apresentadas as possibilidades futuras de sofisticação desta ferramenta.

1. Introdução

Administrar a segurança de um sistema é uma atividade que requer um acompanhamento sistemático dos eventos ocorridos, a fim de se detectar ataques ou tentativas de invasão.

Uma das tarefas mais desafiadoras nesta área consiste em analisar periodicamente os arquivos de *log* do sistema, processo este que envolve a monitoração e análise de vários arquivos, e a correlação dos eventos entre estes.

Embora seja uma atividade extremamente valiosa, tem sido pouco explorada e, em certos ambientes, completamente desprezada.

A maior dificuldade de realizar essa tarefa decorre da imensa quantidade de informação gerada pelos eventos do sistema diariamente. Dependendo da quantidade de tráfego que está sendo monitorado e dos eventos que estão sendo registrados, um único *log* pode conter milhões de entradas por dia. Separar os dados relevantes dos dados comuns é algo que requer muito tempo e atenção.

Sendo assim, desenvolver uma ferramenta para facilitar a análise de *log*, visando a economia de tempo de análise e rapidez na resposta a incidentes, constitui a principal motivação deste trabalho.

Utilizando técnicas de Inteligência Artificial e *'Artificial Ignorance'* tornou-se possível a construção de um sistema baseado em agentes capaz de filtrar as inúmeras entradas de um arquivo de *log*, e produzir um arquivo simplificado, contendo informações relevantes e o número de linhas bem reduzido.

Este documento está organizado do seguinte modo: na seção 2 é apresentada a importância da automação dos logs e os pontos a serem considerados para esta atividade. Na seção 3 é descrita a idéia central da técnica *'Artificial Ignorance'*. Uma visão geral das características de sistemas baseados em agente é esboçada na seção 4. A seção 5 apresenta a descrição, a arquitetura e os resultados obtidos com o sistema desenvolvido neste trabalho. Na seção 6 são apresentadas as conclusões e as idéias para trabalho futuro.

2. Automação da análise de logs

Após revisão e análise manual dos arquivos de *log* por um certo período de tempo, o analista é capaz de entender rapidamente o significado de muitas entradas. Porém, em geral, a maioria das entradas de *log* é de pouco ou nenhum interesse para o observador e o volume de dados é amplo. Convém, então, gerar um relatório somente com as atividades relevantes a serem investigadas. Isto pode ser obtido através da automatização do processo de varredura do *log*.

Criar uma solução para automatizar a análise de *log* pode requerer um tempo significativo, porém, após implementada, ganha-se economia considerável no tempo de análise e resposta a incidentes.
Dois aspectos devem ser levados em consideração ao desenvolver uma ferramenta para automação da análise de arquivos de *log*: o formato e o conteúdo destes arquivos.

2.1. Formato dos arquivos de *log*

Diversas aplicações e sistemas operacionais podem gerar arquivos de *log* de diferentes formatos. Estes arquivos apresentam propriedades específicas que podem ser configuradas explicitamente pelo analista.

Para este trabalho, decidiu-se limitar o escopo de aplicação e considerar a utilização de arquivos de *log* específicos gerados em ambiente *Web server* de formato padrão conhecido.

Cada pedido HTTP de um cliente *Web* resulta em uma entrada (linha) no arquivo de *log*. Um arquivo de *log* é composto de múltiplas entradas, todas com a mesma formatação.

Dentre os formatos de arquivo de *log* HTTP existentes, os que mais se destacam são:

- NCSA *Common Format Log (access log)*
- NCSA *Separate Log Format (three-log format)*
- NCSA *Combined Log Format*
- W3C *Extended Log Format*

O formato-padrão escolhido para este trabalho é o *NCSA Combined Log Format* por ser um padrão estendido de uso comum.

Um exemplo completo de uma entrada em um arquivo deste formato é:

111.134.115.117 - - [08/Jun/1999:08:36:04 +0000] "GET /index.html HTTP/1.0" 404 55 "-" "Mozilla/3.0 [en] (WinNT; I)"

A seguir, são descritos os campos do *log* HTTP de acesso a páginas *Web*, no formato NCSA *Combined Log Format*:

remotehost 111.134.115.117
É o endereço IP ou nome *host/* subdomínio do cliente HTTP que fez o pedido pelo recurso HTTP.

logname -
É um identificador (se disponível) usado para identificar o cliente fazendo o pedido HTTP. Um "-" é usado para indicar que nenhum *logname* está presente.

username -
O *username*, (ou *user* ID) é usado pelo cliente para autenticação quando o recurso HTTP solicitado requer autenticação do usuário. Caso contrário, um "-" é usado para indicar que nenhum *username* está presente.

date [08/Jun/1999:08:36:04 +0000]
Esta é a data e a hora do pedido http, correspondente à data/hora local do servidor.

request "GET /index.html HTTP/1.0"
Este é um pedido HTTP pelo recurso. O campo pedido contém três partes de informação:

- o recurso solicitado - a parte principal (por exemplo, index.html);
- o método HTTP (por exemplo , GET ou POST); e
- a versão do protocolo HTTP do recurso solicitado (por exemplo, HTTP/1.0 ou HTTP/1.1).

status 404
O status é um código numérico de retorno indicando o sucesso ou falha do pedido HTTP.

TABELA 1. Status de pedido HTTP.

100-199	Informativo, indica que a requisição está sendo processada
200-299	Requisição bem-sucedida, o servidor enviará o código HTML sem nenhum problema
300-399	Re-direcionamento do link
400-499	O cliente passou uma requisição incorreta ao servidor, no qual não pôde ser executada
500-599	A requisição foi enviada corretamente, porém o servidor não pôde executá-la por estar com problemas internos.

bytes 55
O campo bytes é um campo numérico contendo o número de bytes de dados transferidos como parte do pedido HTTP, não incluindo o cabeçalho.

referral "-"
É a URL do recurso HTTP que associou o usuário para o recurso que foi requisitado.

agent "Mozilla/3.0 [en] (WinNT; I)"
É a identificação de um cliente HTTP solicitando recursos HTTP. Por padrão , um cliente http geralmente identifica-se pelo nome quando fazendo um pedido HTTP.

cookies "USERID=CustomerA;IMPID=01234"
Cookies são partes de informação que o servidor HTTP envia de volta ao cliente com os recursos solicitados.

Analisando os vários campos existentes e através de aquisição de conhecimento com especialistas na área de segurança em redes, constatou-se que a parte mais vulnerável de um *log* de acesso HTTP é o campo "*request*". E as tentativas de ataque geralmente são provenientes do método GET usado para solicitar recursos.

2.2. Conteúdo dos arquivos de *log*

Existem diferentes tipos de arquivos de *log*, gerados por fontes diversificadas. Os sistemas críticos, tais como, os servidores *Web*, servidores de e-mail, as estações administrativas de sistemas e de banco de dados, sistemas contendo dados corporativos, servidores de bancos de dados e repositórios de código, devem ter os registros históricos de suas atividades armazenados com freqüência.

As atividades normais usualmente registradas consistem de testes de segurança autorizados, erros ou problemas conhecidos e os denominados falsos positivos [3].

Os arquivos de *log* também podem conter informações anormais de eventos de rotina, tais como, assinaturas de ataques ou falhas de sistemas, bem como mensagens impossíveis de serem identificadas.

Os eventos anormais ou ilegítimos podem ser classificados como críticos e não críticos. Varreduras de portas, testes de vulnerabilidades não autorizados, tentativas mal sucedidas de comprometimento dos sistemas são considerados eventos anormais não críticos. Por outro lado, tentativas de invasão bem sucedidas, ataques à redes de parceiros, queda de serviços devido a falhas de hardware ou de software, negação de serviço bem sucedida e mensagens desconhecidas, são considerados eventos críticos.

Algumas assinaturas (padrões não permitidos) conhecidas em *logs* HTTP são: ataques do tipo *worm Nimda*, falha de *hardware* e *port scanning*.

As falhas de hardware geralmente incluem palavras como: *error, traceback, panic, dumping, booting, file system full.*

Quanto às assinaturas de acesso à páginas web, deve-se estar atento para as seguintes entradas [3]:

- Longas cadeias de dados sem sentido (tentativa de buffer *overflow*)
- Tentativas de executar scripts CGI não existentes
- Caracteres especiais submetidos a formulários HTML
- Tentativas de acesso a arquivos de senhas, configurações de *webserver* e ACLs (*access control files*)

Alguns exemplos de atividades ilegítimas são descritos a seguir:

- vulnerabilidades do IIS exploradas pelo *Nimda*, através da chamada aos executáveis root.exe e cmd.exe, conforme exemplo abaixo:

200.46.157.22 - GET /erro/erro404.html 404;http://www/scripts/root.exe?/c+dir 200 - -
03:05:53 200.46.157.22 - GET /erro/erro404.html 404;http://www/scripts/httpodbc.dll 200 - -
03:06:58 200.46.157.22 - GET /erro/erro404.html 404;http://www/scripts/..%%35c../winnt/system32/cmd.exe?/c+dir 200 - -

- Tentativa do *Nimda* de explorar um *backdoor* deixado por um *Code Red II*

204.120.69.195 - - [18/Sep/2001:09:35:12 -0500] "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 - "-" "-"
204.120.69.195 - - [18/Sep/2001:09:35:12 -0500] "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 - "-" "-"

- Tentativa do *Nimda* de explorar o *Unicode* e vulnerabilidades de diretórios

204.120.69.195 - - [18/Sep/2001:09:35:19 -0500] "GET /scripts/..%%35%63../winnt/system32 /cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"
204.120.69.195 - - [18/Sep/2001:09:35:22 -0500] "GET /scripts/..%%35c../winnt/system32 /cmd.exe?/c+dir HTTP/1.0" 400 215 "-" "-"

- Investigação proveniente do *CodeRed*

128.101.47.28 - - [28/Sep/2001:00:43:43 -0400] "GET /default.ida?XXXX ... %u9090%u6858%ucbd3%u7801%u9090%u6858 %ucbd3%u7801%u9090%u6858%ucbd3%u7801 ... 0000%u00=a HTTP/1.0" 404 284 "-" "-"

3. *Artificial Ignorance*

Existem duas maneiras de varrer os arquivos de log para detectar problemas de segurança ou erros. A primeira é procurar por dados conhecidos como indicativos de problema. Este método requer que sejam geradas grandes listas de dados que são considerados de "comportamento suspeito" – listas que são, por sua natureza, incompletas. Além disto, é necessário saber que um problema existe para poder detectá-lo [17].

O outro modo é construir uma lista de dados que representem o comportamento "normal" do sistema, e ignorá-los. Tudo o mais é considerado "fora do ordinário" e toda atenção deve ser voltada para este tipo de informação não esperada.
Esta técnica é denominada "*Artificial Ignorance*" [2] e tem por objetivo eliminar todas as entradas de *logs* que indiquem comportamento normal, disponibilizando as entradas restantes para análise.

4. Agentes

Um software-agente é um programa ou uma função que implementa o mapeamento de percepções em ações [10].

Segundo Juchem [18], um agente é um sistema computacional que está situado em algum ambiente, e que é capaz de realizar ações autônomas neste ambiente visando atingir seus objetivos propostos (vide figura 1).

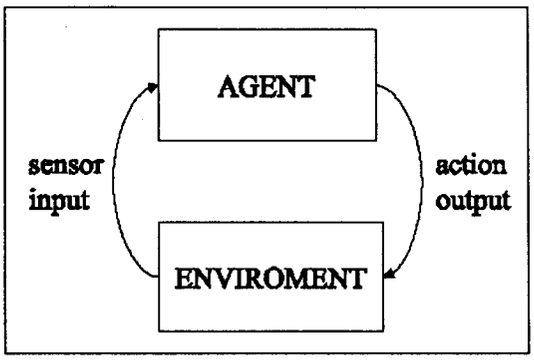

FIGURA 1. Interação Agente-Ambiente.

Em um nível mais alto de abstração, o processamento baseado em agentes pode ser definido como um estado de entrada (o ambiente do agente) que é processado, produzindo um estado de saída (equivalente ao conjunto de efeitos que o agente causa no ambiente).

Os agentes são classificados em geral da seguinte forma: **estáticos** (todo o seu processamento está baseado em um único computador) ou **móveis** (aqueles que têm habilidade de mover-se através de algum tipo de rede de computadores); **reativos** (não têm qualquer tipo de modelo simbólico do ambiente, agindo em resposta a estímulos do ambiente) ou **deliberativos** (processam internamente um modelo de raciocínio simbólico e planejam e negociam com objetivo de coordenarem suas atividades).

Além disso, estas classes de agentes também podem ter outras características, tais como: **autonomia, cooperação** e **aprendizado** [15].

Um agente é **autônomo** quando não é controlado por qualquer entidade externa; um agente é **cooperativo** quando tem habilidade para interagir com outros agentes e/ou pessoas, através de alguma linguagem de comunicação, com o objetivo de atingir seu objetivo ou o do sistema; e finalmente, um agente é **inteligente** quando ele possui uma base de conhecimento, é capaz raciocinar baseado nesta base de conhecimento ou tem capacidade de aprender.

O agente pode ser também categorizado como agente de **tarefa específica** (dedicado a uma tarefa específica ou responsável por conduzir um conjunto de operações de interesse a um usuário ou outro programa), agente de **raciocínio** (raciocina para interpretar percepções, resolver problemas, esboçar inferências e determinar ações), agente **comunicativo** (interage com outros agentes – e possivelmente humanos – através de algum tipo de linguagem de comunicação entre agentes) e agente de **informação** e negociação (ocupa-se em diálogos, negocia e coordena transferência de informações) [16].

Russel e Norvig [11] apresentam uma classificação para arquiteturas de agentes que considera os seguintes tipos: reflexivo simples; reflexivo com estado; baseado em objetivos e baseado em utilidade.

- *Agente Reflexivo Simples* - Nesta arquitetura o agente encontra uma regra cuja condição corresponde à situação atual e executa a ação associada com esta regra.

- *Agente Reflexivo com Estado* - Além das características do agente reflexivo simples, este tipo de agente possui um estado interno que é utilizado para o processo de tomada de decisão e que pode ser atualizado.

- *Agente Baseado em Objetivos* - Em algumas aplicações, conhecer o estado atual do ambiente não é suficiente para saber o que fazer. Isso pode ocorrer quando um problema requer uma seqüência de passos para atingir sua resolução. Neste caso, a busca e o planejamento são sub-áreas da IA que tratam de achar seqüências de ações para atingir objetivos.

- *Agente Baseado em Utilidade* - Um comportamento de alta qualidade não é o resultado de objetivos isolados. Considerando o exemplo de um taxi que pretende chegar a um destino, existem muitas seqüências de ações através das quais se atinge este objetivo, mas alguns são mais rápidos, seguros, mais confiáveis, ou baratos que outros. Objetivos definem tão somente distinções entre o que seja estados "felizes" e "infelizes". O grau de "felicidade" do agente é descrito por uma função que faz o mapeamento de um estado em um número real. O fornecimento de uma especificação completa da função de utilidade permite decisões racionais em dois tipos de caso onde objetivos têm problemas: (1) quando houver objetivos conflitantes, somente um deles pode ser atingido (ex.: velocidade e segurança), a função de utilidade especifica a troca apropriada; (2) quando existem muitos objetivos pretendidos pelo agente, nenhum deles pode ser atingido com certeza, a função de utilidade fornece um caminho na qual a probabilidade de sucesso possa ser pesada frente à importância dos objetivos.

As principais classificações a serem feitas para um ambiente são: acessível ou inacessível; determinístico ou não-determinístico; episódico ou não-episódico; estático ou dinâmico; e discreto ou contínuo [11]:.

- *Acessível x Inacessível* - essa característica determina se o aparato sensorial do agente lhe fornece um estado completo do ambiente. Se isto ocorre, o ambiente é considerado acessível; caso contrário, é considerado inacessível. Um ambiente é considerado efetivamente acessível se os sensores detectam todos os aspectos relevantes para a escolha da ação. No caso de um ambiente ser acessível, não é necessário que o agente mantenha qualquer representação interna do mundo.
- *Determinístico x Não-Determinístico* - se o estado do ambiente é determinado somente por seu estado atual e as atuações dos agentes, podemos dizer que este ambiente é determinístico.
- *Episódico x Não-Episódico* - em um ambiente episódico a experiência do agente é dividida em episódios. Cada um deles consiste em percepções e ações dos agentes, e a qualidade de cada ação depende somente do episódio em si.
- *Estático x Dinâmico* - um ambiente é dinâmico se pode mudar enquanto o agente está deliberando; caso contrário é estático. Um ambiente é dito semi-estático quando não muda com a passagem do tempo, apenas com as ações desempenhadas por agentes.
- *Discreto x Contínuo* - é chamado discreto o ambiente que tem um número limitado de percepções e ações distintas e claramente definidas.

5. O SAAL

O SAAL foi desenvolvido para analisar e filtrar arquivos de *log* de acesso à páginas *Web* de grande volume, identificando as entradas normais, segundo os padrões estabelecidos, que possuem o método GET no campo "*request*" e gravando as entradas restantes, que não casam com o padrão, em outro arquivo para posterior análise.

A partir deste arquivo reduzido, o analista pode identificar com maior precisão as atividades suspeitas e maliciosas.

Este programa baseia-se nos princípios de Inteligência Artificial, mais especificamente na utilização de agentes, e na técnica "*Artificial Ignorance*". Foi implementado em linguagem C++ utilizando a metodologia de orientação a objetos.

O sistema desenvolvido neste trabalho é classificado como agente de tarefa específica, reflexivo simples, estático, reativo e autônomo. É estático, porque tudo acontece dentro de um único computador; é reativo porque os agentes atuam usando um comportamento baseado em resposta a um estímulo; é autônomo porque o sistema não recebe qualquer informação externa depois de ser inicializado.

A tabela a seguir ilustra as características do agente criado.

TABELA 2. Descrição da estrutura do agente desenvolvido.

Agente	Percepção	Ação	Objetivo	Ambiente
Sistema de Apoio à Análise de logs	Leitura de entradas do *log*	Identificar entrada normal	Gravar entrada anormal	Arquivo de *log*

O ambiente utilizado (arquivo de *logs*) é classificado como acessível, não-determinístico, episódico, estático e discreto, conforme apresentado na tabela 3. É **acessível**, pois a leitura das entradas do arquivo dão acesso ao estado total do ambiente. É **não-determinístico**, pois o próximo estado da análise do arquivo de *log* não depende do estado da análise atual e das ações do agente. É **episódico**, pois a experiência do agente é dividida em episódios: (percepção, ação) e o próximo episódio independe do que ocorreu nos anteriores. É **estático**, pois o ambiente (arquivo de *log*) não é alterado enquanto o agente está decidindo o que fazer. É **discreto**, pois há um número limitado de entradas a serem lidas e de ações a serem aplicadas para cada leitura realizada.

TABELA 3. Características do ambiente considerado.

Ambiente	Acessível	Determinístico	Episódico	Estático	Discreto
Arquivo de log	Sim	Não	Sim	Sim	Sim

5.1. Arquitetura

A arquitetura do SAAL, apresentada na figura 2, é baseada na técnica de agente reflexivo simples, e consiste de:

- um sensor responsável pelo disparo da execução do agente (que faz a leitura da entrada – linha - no arquivo de *log*);
- um conjunto de regras (condições) a serem satisfeitas;
- um atuador, responsável pela execução da regra aplicável (escrita da entrada anormal no arquivo) ou não (para o caso de entrada normal).

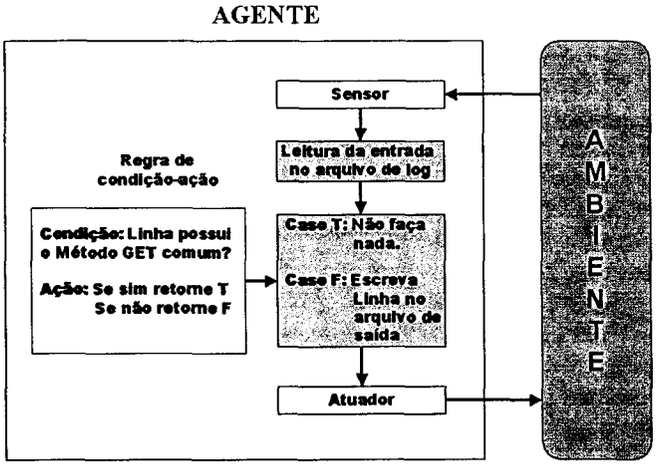

FIGURA 2. Arquitetura do Agente SAAL

A linha do *log* analisada pelo agente será considerada uma entrada normal se atender todas as seguintes regras:

- Linha deve possuir o método GET
- Numero de elementos da linha igual a 9 ou 10
- Após o método GET deve existir um espaço
- Não deve existir a string "../" no campo "*request*"
- Não deve existir comandos *.*exe* no campo "*request*"
- Não deve existir GET /default.ida no campo "*request*"
- Não deve existir mais que 3 elementos no campo "*request*"
- O segundo elemento de "*request*" (recurso solicitado) não deve ser uma string de tamanho maior que 80

5.2. Resultados

Utilizando-se o SAAL, foram analisados três arquivos de *logs* reais extraídos do servidor *web* do LAC/INPE (Laboratório Associado de Computação e Matemática Aplicada) e os resultados obtidos são os apresentados na tabela 4.

TABELA 4. Resultados da aplicação do SAAL.

Arquivo	Tam (linhas) Antes	Tam (linhas) Depois	% (linhas)	Tam (Kb) Antes	Tam (Kb) Depois
Arq1	38483	643	~1,67	7043	122
Arq2	30089	99	~0,32	5671	27
Arq3	29410	70	~0,23	5422	19

6. Conclusão

A partir dos resultados obtidos concluiu-se que:

- O SAAL se mostrou eficiente na identificação de padrões normais.
- A abordagem através de *Artificial Ignorance* mostrou-se aplicável para esse tipo de problema.
- As entradas anormais foram identificadas satisfatoriamente
- As linhas restantes do arquivo gerado refletem as tentativas de invasão no sistema e condições anormais segundo a verificação final feita pelo especialista.

Como idéia para trabalho futuro, tem-se a possibilidade de expansão de regras para incluir as exceções, por exemplo, requisição de arquivos específicos com extensão *.exe* .

Além disto, novos métodos HTTP de solicitação de serviços como, por exemplo, *POST, PROPFIND, TRACE, TRACK*, etc. podem ser acrescentados no código.

A aplicação deste sistema pode ser expandida para identificar padrões de *logs* HTTP nos demais formatos e buscando outros tipos de padrões, bem como a análise de *logs* de aplicações não *Web*.

7. Agradecimentos

Agradecemos ao Dr. Antonio Montes, Prof. Dr. José Demísio Simões da Silva , Cristine Hoepers, Luiz Gustavo Cunha Barbato e a todos que direta ou indiretamente contribuíram para o sucesso desse trabalho.

8. Referências Bibliográficas

[1] HOEPERS, C.; STEDING-JESSEN, K. Análise e Interpretação de *log*. Disponível em: <http://www.nbso.nic.br/docs/palestras/nbso-gter15-tutorial2003.pdf>. Acesso em: 12 abr. 2003.

[2] RANUM, M. Artificial ignorance: how-to guide. Disponível em: <http://archives.neohapsis.com/ archives/ nfr-wizards/1997/09/0098.html>, set. 1997. Acesso em: 02 jun 2003.

[3] HORA, E. C. Análise de Logs: Uma Abordagem Prática. Disponível em: <http://www.tempest.com.br/ attach/ analise-logs-bw.pdf>. Acesso em: 23 jun. 2003.

[4] BIRD, T. Finding the Forest in Your Trees: What Disponível em: <http://www.counterpane.com/log_short.pdf>. Acesso em: 21 jun. 2003.

[5] BIRD, T. Log Analysis Resources and Mailing List. Disponível em: <http://www.counterpane.com/log-analysis.html>, ago. 2002. Acesso em: 13 jun. 2003.

[6] MALICINSKI, A. Monitoring traffic through HTTP server log analysis. Disponível em: <http://www-106.ibm.com/developerworks/web/library/wa-mwt2/?dwzone=web>. IBM web articles, mar 2001. Acesso em 10 jun. 2003.

[7] CERN Web Office, Common Log Format. Disponível em: <http://weboffice-old.web.cern.ch/WebOffice-Old/Services/WWWlogfiles/CommonLogFormat.html>, mar. 2002. Acesso em 13 jun 2003.

[8] NORTHCUTT, S.; et al. **Segurança em Redes.** Rio de Janeiro: Editora Campus, 2002, ISBN 85-3562-1123-3.

[9] NORTHCUTT, S., **Network Intrusion Detection: An Analyst's Handbook.** 2. ed., New Riders Publishing, 2000. ISBN 0735710082.

[10] ROSA, A. Agentes Inteligentes Introdução e Definições básicas. Disponível em: <http://www.urcamp.tche.br/~alexsand/material/ESIH2.html>. Acesso em 11 jun. 2003.

[11] RUSSEL, S.J.; NORVIG, P., **Artificial Intelligence A Modern Approach.** United States: Prentice Hall, 1995. ISBN 0-13-103805-2.

[12] SCHILDT, H. **Inteligencia Artificial Utilizando linguagem C.** São Paulo: McGrawHill, 1989. ISBN: 001.5350285-001.6424

[13] BITTENCOURT, G. **Inteligência Artificial Ferramentas e Teorias,** 2. ed., Florianópolis: Editora da UFSC, 2001. ISBN 85.328.0138-2.

[14] RICH, E.; KNIGHT, K. **Inteligência Artificial.** São Paulo: Editora Makron Books, 1993.

[15]K. FIGUEIREDO, K.; VELLASCO, M. M. B. R.; Pachedo, M. A C. Controle de robô usando agentes inteligentes. I Simpósio Catarinense de Computação - Itajaí, ago 2000.

[16] FRANKLIN, S.; GRAESSER, A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: Proceedings of Third International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag: University of Memphis, 1996.

[17] ABENDSCHAN, J.W. Checksyslog v1.3. Disponível em: <http://www.jammed.com/~jwa/hacks/security/checksyslog/checksyslog-doc.html>. may 2001.

[18] JUCHEM, M.; BASTOS, R.M. Arquiteturas de Agentes. Disponível em: <http://www.inf.pucrs.br/tr/tr013.pdf>, abr 2001. Acesso em: 23 jun. 2003.