Um Sistema de Deteção de Intrusão projetado para o usuário final.

Por

André Fischer
Bel. Em Ciência da Computação (UNILASALLE, 2000)
afischer@ieg.com.br

Vinicius Gadis Ribeiro
MSc (UFRGS,1997), Professor da UNILASSALE e ULBRA
Doutorando PPGC -UFRGS
vribeiro@inf.ufrgs.br

RESUMO

O presente trabalho tem por objetivo apresentar de uma ferramenta de detecção de intrusão desenvolvida recentemente - bem como o seu projeto - que utiliza a tecnologia de agentes para manter a segurança de um sistema computacional. Tal ferramenta fará o monitoramento e diagnóstico de possíveis ataques a computadores que utilizam a plataforma Windows9x e que estejam conectados à *Internet* ou a uma *Intranet*.

Palavras-chave: Sistemas de Detecção de Intrusão, Segurança Computacional, Tecnologia de Agentes Móveis.

ABSTRACT

This paper presents an Intrusion Detection System project that works with agents technology, to guarantee a system computational security. This tool monitors and makes diagnosis of possible attacks to computers that work with Windows 9X computers, connected to Internet or intranet.

***Keywords**: Intrusion Detection Systems, Computer Security, Mobile agents technology.*

1 Ferramentas Intrusion Detection Systems – Sistemas de Detecção de Intrusão

As ferramentas de detecção de intrusão são, na verdade, programas - que não tentam **impedir** a intrusão no momento em que a mesma está ocorrendo, mas apenas procuram **enviar um alerta** ao administrador informando que uma possível violação da segurança está ocorrendo. Opondo-se a essa tendência reativa, observa-se haver estudos que prevêem atitudes pró-ativas, no sentido de impedir a intrusão.

Muitas ferramentas do tipo *Intrusion Detection Systems* (IDS) realizam operações a partir da análise de padrões do sistema operacional e da rede, tais como:

- utilização de CPU;
- E/S de disco;
- uso de memória;
- atividades de usuários;
- número de tentativas de *login*; e
- número de conexões.

Estes dados são atualizados continuamente e formam uma base de informação sobre a utilização do sistema em vários momentos do tempo. Outras ferramentas IDS já possuem bases com padrões de ataque previamente estabelecidos permitindo também a configuração dos valores das bases, bem como a inclusão de novos parâmetros. Com estas informações, uma ferramenta IDS é capaz de identificar tentativas de intrusão e até mesmo registrar a técnica utilizada na mesma, como se fosse uma comparação de impressão digital.

2 Agentes Móveis

Agentes Móveis são um tipo particular de *software* que tem a capacidade de se locomover de uma máquina para outra numa rede. Durante muito tempo eles foram somente um tópico de pesquisa dos laboratórios e a indústria não tinha nenhum interesse nesta tecnologia. Porém, com o desenvolvimento de aplicações para a WWW, a pesquisa na área de Agentes Móveis foi fortemente estimulada.

Agentes Móveis podem ser lançados por *browsers* para procurar informações ou para interagir com qualquer máquina conectada na rede.

Um dos novos paradigmas na criação de ferramentas IDS é o uso da tecnologia dos Agentes Móveis (do inglês *Mobile Agent* - MA). Este novo paradigma nasceu com o lançamento da linguagem JAVA, em 95/96, pela *Sun Microsystems*. Como o JAVA possui um código móvel, foi utilizado como padrão no desenvolvimento de Agentes Móveis.

Contudo, o uso de MAs para detecção de intrusão somente será útil quando muitos, senão todos os *hosts* e dispositivos de rede possuírem uma plataforma que dê suporte aos MAs. Tal plataforma deverá ser simplesmente um *software* de propósito geral que permitirá a implementação de muitas aplicações diferentes.

2.1 Trabalhos relacionados a Agentes Móveis e ferramentas IDS

Existem muitos trabalhos na área de MA e ferramentas IDS. Porém, segundo Jansen [JAN 99], muitos desses ainda estão em fases preliminares, pois não envolvem detecção de intrusão de modo direto, ou não usam a característica de mobilidade dos agentes.

Alguns trabalhos destacados são:

- *Autonomous Agents for Intrusion Detection*: concebido por na Universidade de Purdue [BAL 98], o AAFID é uma aplicação que possui muitas características de uma ferramenta IDS clássica com agentes usados, principalmente, para estruturar a aplicação em módulos mais "leves" e de fácil configuração.
- *Hummingbird*: Desenvolvido pela Universidade de *Idaho* [FRI 98]. Este projeto é um protótipo de ferramenta IDS distribuída mais ambicioso disponível hoje em dia; o mesmo administra abuso no uso dos dados.
- *Java Agents for Meta-Learning*: O projeto JAM [LEE 99], da Universidade de Columbia - Nova Iorque -, aplica meta-aprendizagem num *datamining* distribuído e utiliza agentes inteligentes. Tais agentes empregam técnicas de inteligência artificial para modelar o conhecimento, bem como comportamento, em sociedades multi-agentes ou em domínios.

3 Desenvolvimento da ferramenta IDS

O intuito deste trabalho é desenvolver uma ferramenta IDS que utilize a tecnologia de Agentes para manter a segurança de um sistema conectado em uma rede corporativa. Tal ferramenta é capaz de:

a) monitorar uma máquina conectada na rede, fazendo o papel de "guarda-costas" da mesma, impedindo assim que ela sofra certos tipos de intrusão; e

b) fazer um diagnóstico do que ocorreu ou esta ocorrendo, procurando emitir um aviso para o administrador da rede, ou para o responsável pela segurança da mesma.

Como funcionalidades de uma ferramenta IDS, foram implementadas as seguintes características:

- execução contínua e em segundo plano, sem a supervisão humana;
- monitoramento do sistema;
- resistência a tentativas de mudanças de sua base;
- configuração e atualização com novas técnicas de ataques;
- diagnóstico da intrusão que estiver ocorrendo; e
- aviso automatizado e em tempo real.

As características de Agentes que a ferramenta possuirá serão as seguintes:

- comunicabilidade;
- reatividade;
- habilidade social; e
- persistência.

Tais características decorrem da análise do referencial teórico, pois elas descrevem os requisitos mínimos que uma ferramenta IDS deve conter para poder descobrir uma intrusão e cuidar de sua própria segurança, e também descrevem as características mínimas para a funcionalidade de um Agente.

A ferramenta desenvolvida opera tanto em sistema operacional Windows NT quanto Windows 9x. Porém, o protótipo desenvolvido tem seu uso direcionado para a plataforma Windows 9x, pois é o sistema onde se tem verificado grandes problemas de segurança. Foi colocado como funcionalidade desejável o fácil manejo por usuários finais – típicos desse sistema operacional.

O sistema é orientado a arquivos, isto é, todo o monitoramento e diagnóstico são feitos sobre arquivos selecionados pelo usuário durante a fase de configuração do sistema. Sempre que um novo arquivo for adicionado à base de dados do *software*, este se propagará através do *middleware Common Object Request Broker Architecture* (CORBA) para todas as outras máquinas que possuam a ferramenta IDS, a fim de atualizar suas bases de dados de monitoramento.

A Figura 1 mostra um esquema da ferramenta final após sua implementação.

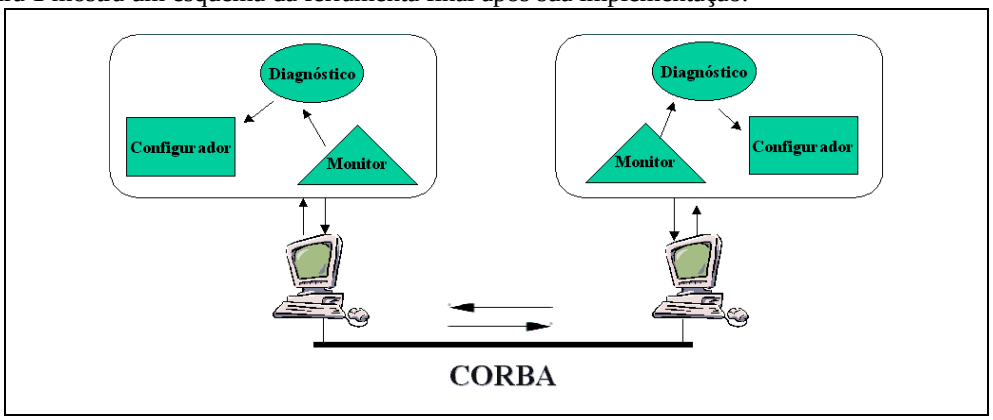

Figura 1: Representação esquemática da ferramenta IDS que foi desenvolvida

A ferramenta disponibiliza, ainda, opções para os usuários escolherem qual tipo de ataque que eles desejam monitorar – como, por exemplo, direitos de leitura, de abertura, de exclusão, de alteração, entre outros.

Uma outra solução para a implementação da ferramenta seria a de centralizar as bases de dados de arquivos a monitorar; porém, esta idéia foi descartada, pois ela demandaria uma grande parte dos recursos do sistema – o que é indesejável para uma ferramenta IDS. Ademais, essa centralização geraria um ponto único de falhas, vindo a comprometer todo o sistema em caso de ocorrência de algum dano nesse ponto.

O software foi desenvolvido em módulos para facilitar sua manutenção e estes módulos estão descritos a seguir.

3.1 O Monitoramento

O monitoramento é a parte mais importante do trabalho, pois é através dele que podemos diagnosticar os acontecimentos e tomar as devidas providências para a solução do problema.

O monitoramento foi implementado através da utilização de módulo de geração de eventos da empresa *AlfaUnits*[1]. Tal gerador de eventos provê muito mais informações sobre o sistema operacional, gerando eventos em quase todas as manipulações de arquivos.

Este gerador de eventos inicia como um serviço no Windows9x e sua execução é realizada em *background*, o que já satisfaz o requisito de uma ferramenta IDS. Ao gerar os eventos, o monitor preenche uma estrutura que é capturada como uma mensagem pelo módulo diagnóstico.

A estrutura da mensagem do gerador de eventos se encontra no quadro 1.

```
Type
   TlogEntry = record
      dwOperation : DWORD;
      dwResult    : DWORD;
      dwProcessLen: DWORD;
      dwFileLen   : DWORD;
      dwDestLen   : DWORD;
      szProcess   : array[ 0..MAX_PROC - 1 ] of char;
      szFileName  : array[ 0..MAX_PATH - 1 ] of char;
      szDestName  : array[ 0..MAX_PATH - 1 ] of char;
   end;
   PLogEntry = ^TLogEntry;
```

Quadro 1: Estrutura da mensagem disparada pelo monitoramento.

3.2 O Diagnóstico

O diagnóstico é feito avaliando-se os eventos gerados pelo monitor. Através da estrutura exibida no quadro 2, o módulo de diagnóstico consegue descobrir que tipo de evento ocorreu e qual processo executou tal evento. Para tanto, precisamos ativar o diagnóstico, para que o mesmo possa executar sua tarefa.

A cada evento gerado, uma mensagem é disparada e o diagnóstico trata esta mensagem através da rotina do quadro 2:

```
procedure Tdiagnostico.WndProc;
begin
   if ( Msg.Msg = hAlfaMessage ) and FActive then
      CapturaEvento
   Else
      Msg.Result := DefWindowProc( FHwnd, Msg.Msg, Msg.wParam,
         Msg.lParam );
end;
```

Quadro 2: Procedimento executado cada vez em que um evento é gerado pelo monitor

Toda a mensagem que seja igual a previamente definida no diagnóstico faz com que a rotina *CapturaEvento* seja executada. É no bloco principal deste procedimento que as mensagens são decodificadas. O quadro 3 mostra o bloco principal que decodifica as possíveis intrusões que podem ocorrer durante o uso do sistema.

O campo *dwOperation* da mensagem enviada pelo monitor possui a operação executada pelo "provável intruso" e é através deste campo que são disparados os eventos para que a resposta a intrusão possa ser gerada.

```
Case dwOperation of
   IFS_RENAME : if ( dwResult = 0 ) and Assigned( FOnRename ) then
                   FOnRename( Self, szProcess, szPath, szExtra );
   IFS_OPEN   : if ( dwResult = 0 ) and Assigned( FOnOpen ) then
                   FOnOpen( Self, szProcess, szPath );
   IFS_DELETE : if ( dwResult = 0 ) and Assigned( FOnDelete ) then
                   FOnDelete( Self, szProcess, szPath );
   IFS_READ   : if ( dwResult = 0 ) and Assigned( FOnRead ) then
                   FOnRead( Self, szProcess, szPath );
   IFS_WRITE  : if ( dwResult = 0 ) and Assigned( FOnWrite ) then
                   FOnWrite( Self, szProcess, szPath );
   IFS_REPLACE: if ( dwResult = 0 ) and Assigned( FOnReplace ) then
                   FOnReplace( Self, szProcess, szPath );
End; //fim case dwOperation
```

Quadro 3: Bloco que decodifica possíveis ameaças

Para podermos utilizar tanto o monitoramento como o diagnóstico é necessário uma *interface* onde as características de cada um dos módulos possam ser configuradas adequadamente. Neste sentido, faz-se necessário a implementação do módulo de configuração que será visto no próximo item.

3.3 A Configuração

É através da *interface* provida pelo módulo de configuração, que são ativados tanto o monitoramento quanto o diagnóstico. Com ele, podemos indicar quais arquivos monitorar, quem avisar no caso de uma intrusão e quais são as

[1] Endereço eletrônico em setembro de 2000: *http://www.alfaunits.com*

possíveis intrusões a serem monitoradas. É o módulo de configuração que também provê a resposta à intrusão, quando constatada.

3.3.1 Interface do Módulo de Configuração

A *interface* deste módulo foi desenvolvida de modo a facilitar as escolhas das intrusões a serem detectadas e facilitar também a escolha dos arquivos a serem monitorados.

A Figura 2 mostra a tela principal do módulo de configuração.

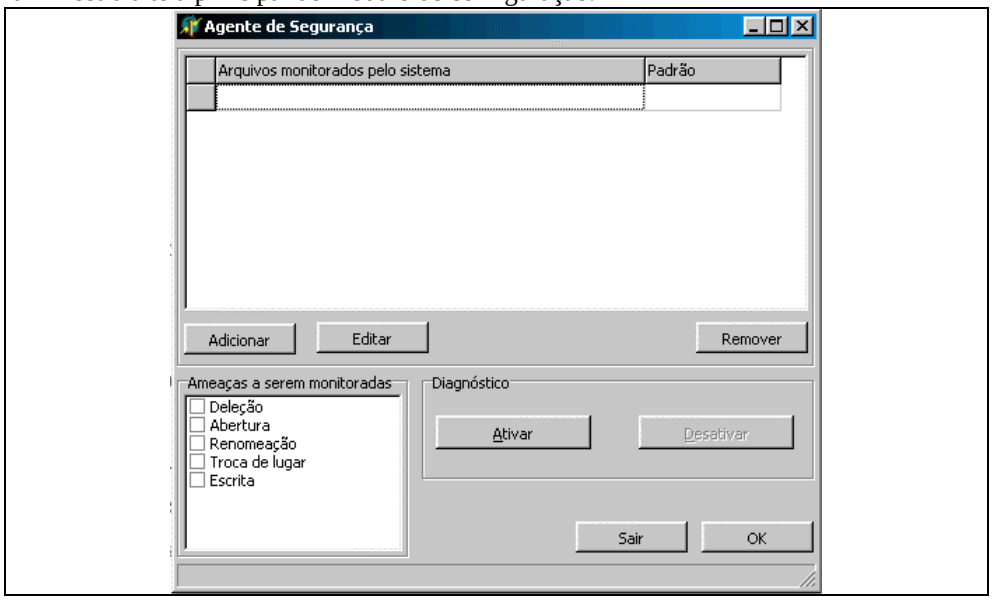

Figura 2: *Interface* do módulo de configuração com o usuário

3.4 Interação da Ferramenta IDS com a máquina

A ferramenta entra em ação logo após o término de sua instalação. Primeiro, o monitoramento é iniciado e se instala como se fosse um serviço do Windows; depois, o usuário ativa o diagnóstico através da tela de configuração.

Na primeira vez que o programa entra em uso, a sua localização é adicionada ao banco de dados, onde são armazenadas as informações sobre os arquivos a serem monitorados. Isto faz com que ele passe a se automonitorar, a fim de manter a sua integridade.

A primeira máquina da rede a ter instalado o *software* será eleita como servidora dos objetos CORBA. Este servidor possui um objeto denominado de "mensageiro", que atualiza as informações de monitoramento das máquinas e envia os alertas para as pessoas a serem avisadas. É através dele que é realizado o monitoramento da rede. Para que este monitoramento ocorra, é necessário que um programa chamado *SmartAgent* – que acompanha o pacote CORBA – esteja em execução, pois é este programa que faz com que as requisições dos *softwares* clientes cheguem e retornem do *software* servidor.

A figura 3 exemplifica a interação entre os módulos:

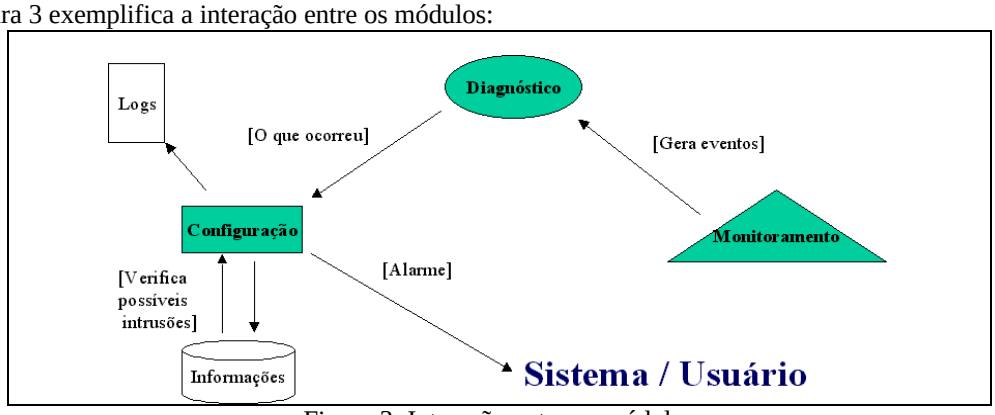

Figura 3: Interação entre os módulos

O objeto mensageiro possui dois atributos importantes, chamados *Atualizar* e *Alertar*. Estes campos são verificados periodicamente pelas ferramentas IDS. No momento em que um novo arquivo é acrescentado na base de dados de monitoramento de um dos *softwares*, o valor do campo *Atualizar* muda de "falso" para "verdadeiro". Isto provoca uma atualização em todos os outros Agentes de Segurança. É a própria ferramenta IDS que se encarrega de atualizar sua base de dados, utilizando, para isto, o objeto mensageiro através do seu método *AtualizarInfo*. O quadro 4 mostra como o atributo *Atualizar* é verificado, e qual é a chamada que é feita ao objeto mensageiro para a atualização da base de dados de arquivos monitorados.

```
Var
   ObjMsg: Imensageiro; // instaciamento do objeto mensageiro
...;
...;
begin
   ...;
   ...;
   // verificação do atributo "atualizar"
   if objMsg.Atualizar then
      // atualização do banco de dados de monitoramento
      objMsg.AtualizarInfo;
   ...;
end;
```

Quadro 4: Verificação do atributo Atualizar e Atualização do banco de dados de monitoramento

No caso de ocorrer uma intrusão, o atributo *Alertar* é quem sofre a modificação alterando seu valor de "falso" para "verdadeiro". Isto faz com que a ferramenta IDS "pergunte" ao sistema quem está utilizando a máquina. Se for o usuário a ser avisado, então uma mensagem é apresentada na tela do usuário, caso não seja, nada ocorre.

O quadro 5 mostra como é feita a verificação do atributo *Alterar* do objeto mensageiro.

```
Var
   vQuemAvisar: string;
   vMensagem: string;
begin    // verificação do atributo "alertar"
   if objMsg.Alertar then begin
      // atualização do banco de dados de monitoramento
      objMsg.EnviarAlerta( vQuemAvisar, vMensagem );
      if ( vQuemAvisar = GetUserName ) then
         MessageDlg( vMensagem, mtWarning, [ mbOK ], 0 );
   End;
end;
```

Quadro 5: Verificação do atributo Alertar e envio da mensagem de alerta de intrusão

São criados arquivos de *log* cada vez que uma intrusão ocorre, e neste arquivo, são armazenados:

- quais arquivos atacados geraram os alertas;
- que processo realizou a intrusão; e
- de qual máquina partiu tal ataque.

Também são feitos *checkpoints* - no caso, marcações feitas em um arquivo com a estrutura hora, data, arquivos monitorados, a quem avisar e "gravado com sucesso" - de dez em dez minutos, que visam manter a persistência dos agentes envolvidos no monitoramento e diagnóstico.

Na detecção de uma intrusão, ocorrem os procedimentos conforme os seguintes passos:

1º) um evento é gerado pelo monitor que passa a mensagem para o módulo diagnóstico;

2º) o módulo diagnóstico captura este evento e "traduz" esta mensagem que é então ao módulo de configuração na forma, também, de um evento;

3º) o módulo de configuração faz a verificação na sua base de dados, para verificar se o arquivo que sofreu a ação é um dos que estão sendo monitorados; caso seja, o módulo dispara uma mensagem para a pessoa configurada previamente para receber o aviso de intrusão.

A figura 4 demonstra um exemplo da seqüência das *interfaces* ativadas por ocasião da tentativa de se apagar um arquivo monitorado.

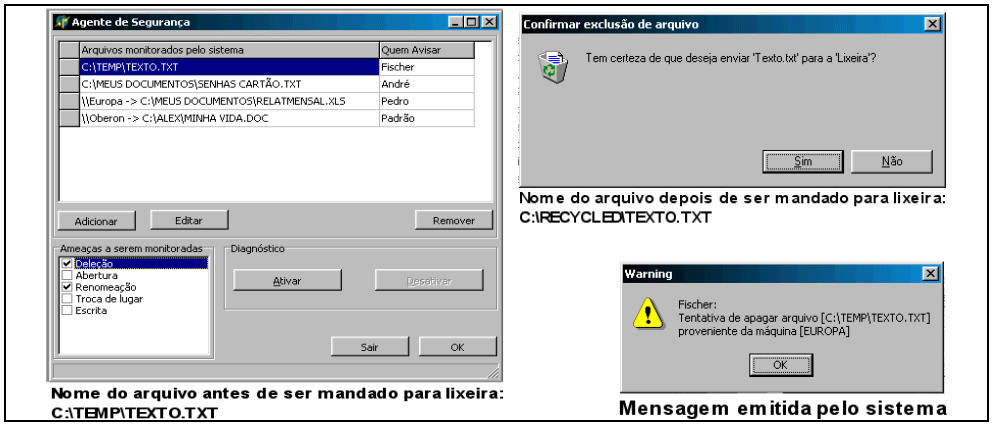

Figura 4: Seqüência das *Interfaces.*

Primeiro, o nome do arquivo é acrescentado na listagem de monitoramento; logo após, manda-se o arquivo para a lixeira. Ao fazer isto, o nome completo do arquivo é alterado de C:\TEMP\TEXTO.TXT para C:\RECYCLED\TEXTO.TXT; isto faz com que o *software* entre em ação e reporte a ameaça.

4 Conclusão

Estamos no início do desenvolvimento de ferramentas IDS realmente inteligentes. Ainda temos muitos problemas com estas ferramentas, tais como os falsos positivos e falsos negativos. Contudo, temos o exemplo dos primeiros antivírus do mercado que consideravam qualquer novo arquivo criado como sendo contaminado, parando o funcionamento do computador (pois o programa julgava que um vírus estava agindo). Atualmente, estes programas antivírus estão bem mais desenvolvidas e muitas vezes nem se toma conhecimentos delas - mas as mesmas estão presentes, até na forma de serviços. Com as ferramentas IDS ocorrerá o mesmo, pois o uso de agentes inteligentes tornará, num futuro próximo, os MAIDS capazes de deter grande parte das intrusões, sem que os usuários sequer tomem conhecimento disso.

Como trabalhos futuros, poderíamos destacar a inclusão de um módulo de auto-autenticação na ferramenta IDS utilizando criptografia. Isto diminuiria as chances de ocorrerem falsos negativos no momento da utilização do *software*. Com esta autenticação, seria possível incluir uma reatividade maior na ferramenta (podendo até fazer com que ela interrompesse uma intrusão em andamento). Outra alteração interessante seria a inclusão do *e-mail* da pessoa a ser avisada, pois se a mesma não se encontra-se autenticada na rede, um *e-mail* poderia ser enviado para ela informando sobre a intrusão ocorrida. Também seria interessante disponibilizar-se a ferramenta de modo a possibilitar a utilização em grupos de trabalho. Tal facilidade seria útil no momento de se trabalhar com arquivos compartilhados, pois a ferramenta identificaria que as pessoas que estariam utilizando tais arquivos fariam parte do mesmo grupo, evitando o falso positivo quando do uso por alguém do grupo. Pode-se permitir, como opção *default*, o informe ao administrador do sistema. Um outro trabalho que se encontra em andamento é a constituição de um banco de modelos de intrusão, para uso futuro.

No que diz respeito aos problemas encontrados durante a o desenvolvimento da aplicação, a maior dificuldade ocorreu quando se desenvolveu o módulo de monitoramento do sistema. Devido a falta de documentação de funções ou classes que permitissem a execução do mesmo, não foi possível implementá-lo de modo direto - tamanha a complexidade envolvida na sua codificação. Há funções que realizam o monitoramento, mas estas não possuem todas as funcionalidades necessárias. Para realizarmos um monitoramento confiável, necessitamos trabalhar com uma programação baixo nível, onde precisamos realizar alterações em arquivos do tipo .VXD e .SYS.

Referências Bibliográficas

[BAL 98] BALASUBRAMANIYAN, Jai, GARCIA-FERNANDEZ, José Omar, ISACOFF, David, SPAFFORD, E. H. e 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>. Acessado em abr. 2000.

[CIDF98] **Common Intrusion Detection Framework Specification**, Versão 0.6.
Disponível em: <http://seclab.cs.ucdavis.edu/cidf/>. Acessado em abr. 2000.

[DES 91] DESWARTE, Yves, BLAIN, Laurent e FABRE, Jean-Charles, **Intrusion Tolerance in Distributed Computing Systems**, published in Proc. of the IEEE Symposium on Research in Security and Privacy, maio de 1991.
Disponível em: <http://www.researchindex.com>. Acessado em abr. 2000.

[FRI 98] FRINCKE, D., TOBIN, Don, McCONNELL, Jesse, MARCONI, Jamie e POLLA, Dean, **A Framework for Cooperative Intrusion Detection,** 21st National Information Systems Security Conference, pág. 361-373, outubro de 1998.
Disponível em: <http://csrc.nist.gov/nissc/1998/papers.html>. Acessado em maio 2000.

[JAN 99] JANSEN, Wayne; MELL, Peter; KARYGIANNIS, Tom e MARKS, Don, **Applying Mobile Agents to Intrusion Detection and Response**, Nation Institute of Standards and Technology, outubro de 1999.
Disponível em: <http://www.researchindex.com>. Acessado em maio 2000.

[KUM 95] KUMAR, Sandeep, **Classification and Detection of Computer Intrusions**, agosto de 1995.
Disponível em: <http://www.researchindex.com>. Acessado em maio 2000.

[LEE 99] LEE, W., STOLFO, S. J. e MOK, K., **A Data Mining Framework for Building Intrusion Detection Models,** IEEE Symposium on Security and Privacy, 1999.Disponível em: <http://www.cs.columbia.edu/~sal/JAM/PROJECT/>. Acessado em abr. 2000.

[NED 99] NED, Frank, **Ferramentas de IDS**, setembro de 1999.
Disponível em: <http://www.rnp.br/newsgen/9909/ids.shtml>. Acessado em abr. 2000.

[WOO 94] WOOLDRIDGE, Michael e JENNINGS, Nick, **Intelligent Agents: Theory and Practice,** 1994.
Disponível em: <http://www.doc.mmu.ac.uk>. Acessado em abr. 2000.