Módulos de Monitoramento para IDS híbrido

Mateus Fernandes Dornelles
Bel em Ciência da Computação
UNILASALLE(2001)
mfd@ieg.com.br

Vinicius Gadis Ribeiro
Professor da ULBRA e
UNILASALLE
Doutorando do PPGC/UFRGS
vribeiro@inf.ufrgs.br

Raul Fernando Weber
Professor do PPGC/UFRGS
weber@inf.ufrgs.br

O presente trabalho apresenta os módulos de monitoramento, os quais fazem parte de um projeto de um sistema de detecção de intrusão híbrido em andamento - o qual utiliza a tecno*log*ia de agentes e agentes móveis. Tal sistema fará o monitoramento, diagnóstico e a tomada de ações contra possíveis ataques a uma determinada máquina ou a própria rede de computadores que utilizarem a plataforma Linux e que estejam conectados à *Internet* ou a uma *Intranet*, além de ser tolerante a falhas.
Palavras-chave: segurança computacional, sistemas de detecção de intrusão, agentes e agentes móveis.

Abstract
This paper presents the monitoring modules that are part of a project of a hybrid intrusion detection system in progress that uses technology of agents and mobile agents. Such system will monitor, diagnose and take actions against possible attacks to a certain machine or a computer net itself that uses Linux or those which are connected to Internet or an intranet, being fault tolerant.
Keywords*: computer security, intrusion detection systems, agents and mobile agents.*

1 Política de Segurança

Segundo Cerias [CER 01], uma política de segurança define o que pode ser permitido e o que deve ser proibido em um sistema. Temos basicamente duas filosofias, que podem ser utilizadas para escrever qualquer política de segurança. Uma política de segurança deve descrever exatamente quais são as operações permitidas em um sistema. Qualquer operação que não esteja descrita de forma detalhada na política de segurança, deve ser considerada como um ação ilegal ao sistema. Estas filosofias podem ser denominadas de Proibitiva e Permissiva.

Na filosofia Proibitiva, tudo que não é expressamente permitido é proibido. Como exemplo de uma política de segurança proibitiva, pode-se citar um laboratório de Informática que utiliza Windows NT, onde os usuários não tenham permissão, por exemplo, para definir a hora local, visto esta não foi setada. Nesse caso, os usuários só têm permissão para fazer o que for expressamente permitido, sendo proibida a instalação de softwares que necessitem fazer alterações no registro do sistema.

Já na Permissiva, tudo que não é expressamente proibido é permitido. Como exemplo deste tipo de política de segurança, pode-se citar um laboratório onde as normas que estão escritas no papel podem ser burladas como, por exemplo, a instalações de softwares piratas e o uso de programas de *chat*.

Da definição da política de segurança é que será definida a forma de trabalhar de um IDS.

1.1 Elementos de uma política de segurança

Um sistema de computadores (rede, aplicativos, IDS...) pode ser considerado como um conjunto de recursos que é disponibilizado para ser utilizado pelos usuários autorizados. Segundo [CER 01] e [NED 99], uma política de segurança pode ser descrita em seis elementos, apresentados a seguir:

- Disponibilidade - o sistema deve estar disponível para ser utilizado quando o usuário necessitar. Dados críticos devem estar disponíveis de forma ininterrupta.
- Utilização - os sistemas e os dados devem ser utilizados para as devidas finalidades.

- Integridade - os sistemas e os dados devem estar completamente íntegros e em condições de serem acessados a qualquer momento.
- Autenticidade - o sistema dever ter condições de verificar a identidade do usuário e o usuário deve ter condições de verificar a identidade do sistema.
- Confidencialidade - dados privados devem ser apresentados somente para os donos dos dados ou para o grupo de usuários autorizados.
- Posse - o dono do sistema deve ter condições de controlá-lo.

2 Intrusos e Intrusões

Dentro de um Sistema de Detecção de Intrusões, podemos destacar dois aspectos fundamentais: os intrusos e as intrusões. Segundo Cerias [CER 01], um intruso pode ser caracterizado através de uma boa política de segurança, que deve ser previamente definida. Eles podem ser classificados como segue abaixo:

- Intrusos Externos - Atacam de fora do sistema, geralmente da Internet.
- Intrusos Internos – Atacam de dentro da instituição que possui o sistema.

A maioria dos usuários imagina que os ataques se originam de fora de sua instituição. Mas de acordo com [NED 99], geralmente a maior porcentagem de ataques tem origem de dentro do próprio sistema (por exemplo, um funcionário descontente). Afinal, quem conhece melhor a topo*log*ia da sua rede? Quem sabe onde os dados importantes estão armazenados e quais são os recursos de segurança disponíveis? Por isso as pessoas que têm muito contato com o sistema, devem ser constantemente investigadas e questionadas.

As intrusões podem ser caracterizadas como um conjunto de ações que tentem comprometer a integridade, confiabilidade ou disponibilidade de dados e/ou sistema [KUM95]. Porém, em relação a classificação das intrusões, existem algumas variações bibliográficas, segundo Kummar [KUM 95], Cerias [CER 01], as intrusões podem ser divididas em duas classificações diferentes, descritas a seguir:

- Intrusão devido a mudança de padrão (*anomaly*) - são detectadas com a observação de mudanças de uso em relação ao padrão normal do sistema. Primeiro, monta-se um perfil do sistema; em seguida, através de monitoração, procura-se por divergências significantes em relação ao perfil construído. Por exemplo, se for detectada, atividade na conta do usuário X - que usa um computador em seu local de trabalho entre 9 e 17 horas - após 17 horas, pode ser considerado como uma anomalia e pode vir a ser uma intrusão. Se outro usuário Y sempre se conecta a um determinado servidor fora do horário de trabalho, a conexão a outro *host* da rede pode ser considerada anormal.
- Intrusão devido ao mau uso do sistema (*misuse*) - são os ataques realizados a pontos fracos do sistema, pontos estes conhecidos. Eles podem ser detectados a partir da monitoração de certas ações realizadas em determinados objetos. Por exemplo, um ataque de *worms* de *Internet* ocorrido em 1989 que fez uso dos *bugs* presentes no *softwares finger* e *sendmail*, podem ser citados nesta classificação. Este tipo de intrusão, faz uso direto de vulnerabilidades do sistema.

3 Sistemas de Detecção de Intrusões

Uma forma de se defender de intrusões é fazer uso de Sistemas de Detecção de Intrusões (do inglês IDS - *Intrusion Detection System*). O primeiro conceito de IDS foi proposto por James Anderson, em 1980. Porém, somente em 1987, quando Dorothy Denning publicou um modelo de detecção de intrusão, foi que esta área começou a incrementar esforços em pesquisas. Em

1988, foram criados três protótipos de IDS: um por David Bauer e Michael Koblentz, um por Michael Sebring e outro por Stephen Smaha, sendo que um grande número de protótipos foram criados nos anos seguintes [FIS 01].
Atualmente, as ferramentas de IDS são usadas em quase todas as redes de grandes corporações, governos e universidades, que preocupam-se com a segurança de seu sistema. A área de detecção de intrusão é uma área comercial bem estabelecida, com grandes competidores, tais como *Cisco* e *Network Associates,* entre outras grandes empresas.
Muitas destas ferramentas funcionam analisando padrões do Sistema Operacional e da rede, como utilização de Unidade de Processamento Central (CPU), entrada e saída de dados, uso de memória, *logs* dos usuários, número de conexões, volume de dados trafegando em um segmento de rede, etc. Este sistema pode formar uma base de dados sobre a utilização do sistema em vários tempos. Outras já trazem base de dados com padrões previamente definidos, com possibilidade de inclusão de novos padrões.
De acordo com as bases de dados, o sistema IDS pode identificar uma tentativa de invasão, e até procurar identificar a técnica de invasão utilizada. Conforme [Cerias 00] os Sistemas de Detecção de Intrusos (IDS) podem ser divididos em três categorias principais :

- IDS de Rede - seu funcionamento básico se dá através da captura de todos os pacotes que trafegam pela rede através de um ou mais sensores. Eles mandam as informações coletadas para um console que irá analisá-las. Um IDS de rede pode incluir mecanismos que possam verificar a integridade de arquivos importantes no sistema;
- IDS Local - analisa os pacotes que são enviados para a máquina na qual reside. Pode funcionar independentemente ou enviar relatórios para um sistema mestre;
- IDS Híbrido - combina um IDS de rede com um IDS local.

Conforme [CER 01], [NED 99] um sistema IDS deve possuir algumas características fundamentais, dentre as quais podemos destacar :

- deve rodar continuamente sem interação humana e deve ser seguro o suficiente de forma a permitir sua operação em *background,* mas não deve ser complexo;
- deve ter tolerância a falhas, de forma a não ser afetado por uma falha do sistema, ou seja, sua base de dados não deve ser perdida quando o sistema for reinicializado;
- deve resistir a tentativas de mudança (subversão) de sua base, ou seja, deve monitorar a si próprio de forma a garantir sua segurança;
- deve ter o mínimo de impacto no funcionamento do sistema;
- deve detectar mudanças no funcionamento normal;
- deve ser de fácil configuração. Cada sistema possui padrões diferentes e as ferramentas de IDS devem ser flexíveis aos diversos padrões.
- deve cobrir as mudanças do sistema durante o tempo, como no caso de uma nova aplicação que se integre ao sistema.

Além destas características, é fundamental que o sistema seja capaz de reconhecer os prováveis erros que possam ocorrer no sistema. Estes erros podem ser divididos em três categorias, que são :

- Falso Positivo: ocorre quando a ferramenta classifica uma ação como uma possível intrusão, quando na verdade trata-se de uma ação legítima.

- Falso Negativo: ocorre quando uma intrusão real acontece mas a ferramenta permite que ela passe como se fosse uma ação legítima.
- Subversão: ocorre quando o intruso modifica a operação da ferramenta de IDS para forçar a ocorrência de falso negativo.

Contudo, para a composição de um Sistema de Detecção de Intrusão(em inglês, *Intrusion Detection Systems*, ou IDS), diversos elementos constituintes devem trabalhar cooperativamente. Dentre esses, destacam-se os módulos de monitoramento - necessários para efetuar o acompanhamento das operações de um IDS.

4 Os Módulos desenvolvidos

Considerando-se a relevância que os IDS têm para a segurança computacional, tem-se observado a tendência crescente de trabalhos de Instituições acadêmicas e comerciais. O recente trabalho de Fischer[FIS 01] apresentou um IDS orientado a usuários finais em ambiente Windows 9X, monitorando arquivos previamente escolhidos, empregando a tecno*log*ia de agentes. Contudo, a necessidade de IDS mais robustos, os quais monitores atividades suspeitas em outros ambientes - tais como em redes de computadores - incorre em projetá-los de outra maneira[CAM 01]. Considerando-se a necessidade básica do monitoramento e do diagnóstico, o presente trabalho enfocou a implementação esses módulos - tanto em clientes, quanto para os servidores dessa rede -, permitindo o posterior desenvolvimento incremental de um IDS.

A seguir, são apresentadas algumas considerações referentes aos módulos desenvolvidos.

4.1 O ambiente de Desenvolvimento

Os módulos foram desenvolvidos sobre a plataforma Linux/x86. A distribuição de Linux utilizada foi o Red Hat 7.1 (*SeaWolf*), por esta ser a distribuição mais amplamente utilizada. Já a linguagem de programação escolhida foi o C++, usando o compilador GCC da Gnu, devido a seu excelente desempenho e portabilidade entre diversas plataformas. Foram também utilizados alguns *scripts* de inicialização *bash* e algumas bibliotecas para auxiliar o desenvolvimento do protótipo, destacando-se entre elas a *libnids*, *libpcap*, e a *libnet*. Todos os arquivos de *logs* gerados pelo uso dos módulos são gravados em arquivos de registros.

Foram utilizadas duas máquinas (Pentium 233 MMX com 64 MB de RAM cada uma) para o desenvolvimento do protótipo.

4.2 O protocolo RPC (Remote Procedure Call)

O RPC (*Remote Procedure Call*) é um protocolo que provem o paradigma de comunicação de alto-nível usado em sistemas operacionais. O RPC pressupõe que tenha sido previamente instalado um protocolo de comunicação de baixo nível, como o TCP/IP(*Transmission Control Protocol/Internet Protocol*) ou UDP (*User Datagram Protocol*), para transmitir os pacotes de dados entre as aplicações que se comunicam. O RPC implementa um sistema de comunicação lógica entre cliente-servidor, projetado especificamente para trabalhar sobre redes.

O protocolo RPC foi constituído sobre o protocolo XDR (*eXternal Data Representation*), que define os padrões de troca de dados em aplicações que se comunicam remotamente. O XDR converte os parâmetros e resultados para cada requisição RPC.

Esse protocolo permite que usuários possam trabalhar com procedimentos remotos, como se estes fossem procedimentos locais. Por exemplo, um usuário na máquina A pode solicitar que a máquina B execute a função soma (a,b) nela existente e esta, então, repassa o resultado novamente para a máquina A. As chamadas de procedimentos remotos são definidas através de rotinas existentes no protocolo RPC. Para cada requisição existe uma mensagem de

retorno. O protocolo RPC é um protocolo de troca de mensagens que implementa outros protocolos como, por exemplo, chamadas de procedimentos remotos *broadcasting*.
Um cliente é uma máquina ou processo que acessa os serviços ou recursos de outra máquina ou processo em uma rede. Servidor é um computador que provê acesso a serviços ou recursos e implementa serviços de rede. Cada serviço de rede é uma série de programas remotos. Cada programa remoto implementa uma série de procedimentos remotos.
O protocolo RPC provê um mecanismo de autenticação que identifica o servidor e o cliente para ambos, sendo suportados vários tipos de sistemas de criptografia - como o DES, por exemplo (*Data Encryption Standard*).No protocolo RPC, cada servidor executa um programa que é um conjunto de procedimentos remotos. A combinação de endereço do servidor, numero do programa e do procedimento identificam uma *procedure* remota. Quando um pacote com uma requisição chega ao servidor, aciona um função que executa o procedimento requisitado e retorna outro pacote com a requisição feita.
A interface do protocolo RPC é geralmente usada para a comunicação entre processos de máquinas remotas. Contudo, o protocolo RPC trabalha da mesma forma com diferentes processos/programas rodando na mesma máquina.
O protocolo RPC foi usado por suprir todas as necessidades do protótipo desenvolvido e já disponibilizar um bom suporte nativo na distribuição de Linux utilizada. Contudo, uma nova versão desses módulos encontra-se em desenvolvimento - o qual está empregando *sockets*.

4.3 As bibliotecas utilizadas

A biblioteca *libnet* é uma coleção de rotinas para auxiliar a construção e tratamento de pacotes que circulam pela rede. Ela provê um *framework* para trabalhar a baixo nível com interceptação de pacotes de rede, tratamento de pacotes e injeção de novos pacotes na rede. A biblioteca *libpcap* implementa a mesma funcionalidade da biblioteca *libnet* - apenas que a um nível mais alto.
A biblioteca *libnids* implementa um componente de Sistema de Detecção de Intrusões de Redes. Ele emula o IP *stack* do Linux 2.0.x. O *libnids* oferece desfragmentação IP, construção de pacotes TCP e detecção de TCP *port scan*.

4.4 A arquitetura do sistema de monitoramento

O sistema de monitoramento é divido em três módulos: Módulo de Monitoramento de Clientes, Módulo de Monitoramento de Redes e Módulo Servidor. O módulo de monitoramento de clientes e o módulo de monitoramento de redes, depois de instalados, executam sem interação com usuário. Ambos os módulos geram arquivos de *logs* locais os quais, mais tarde, podem ser analisados pelo Módulo Servidor. Como já foi dito anteriormente, o protótipo foi desenvolvido em duas máquinas; numa delas o módulo servidor e o módulo de monitoramento de redes eram executados e na outra, o módulo cliente era executado. A figura 1, a seguir, apresenta a arquitetura do protótipo desenvolvido.

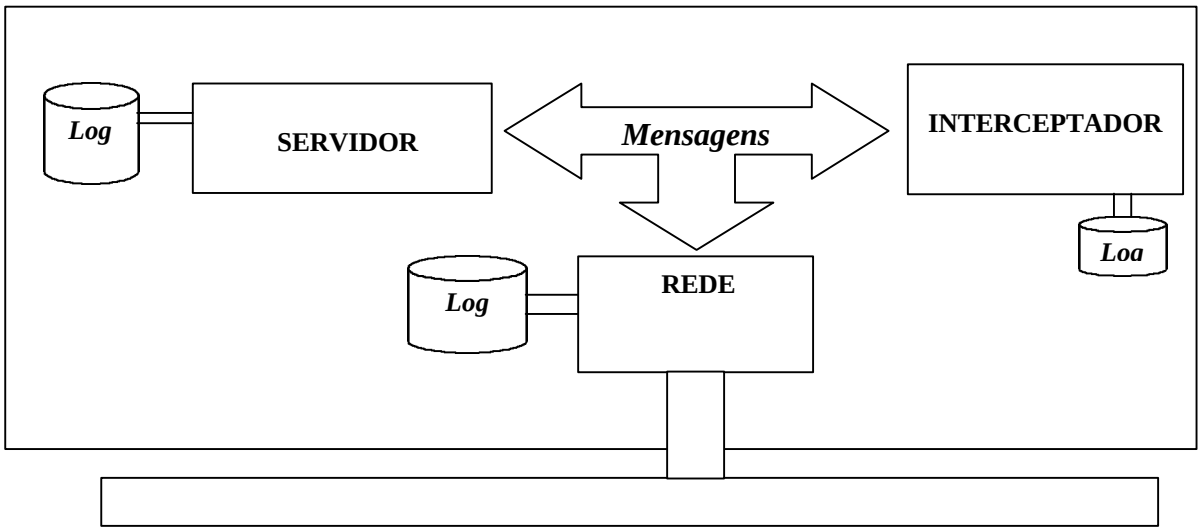

Figura 1 – Arquitetura do protótipo desenvolvido

4.5 O módulo de Monitoramento de clientes

O Módulo de Monitoramento de Clientes, que é uma extensão do kernel do Linux e pode ser considerado um agente, funciona capturando as chamadas de sistema que o usuário ou processos executam. Na verdade, cada módulo cliente atua de modo similar a um "servidor" e o módulo servidor funciona como se fosse um cliente desse "servidor". O quadro a seguir mostra uma parte do código que realiza a interceptação das chamadas que ocorrem no sistema.

```
Extern void* sys_call_table[];/*Usado para ter acesso a tabela
com as chamadas de sistema*/
 ...
int (*orig_mkdir)[1](const char *path); /*chamada de sistema original*/
...
int meu_mkdir(const char *path)
{
...
Save_Log (“mkdir”,path);
  SYS_mkdir(path);
}

int init_module(void)                /*Inicialização do modulo*/
{
...
 orig_mkdir=sys_call_table[SYS_mkdir];
...
return 0;
}

void cleanup_module(void)            /*Finalização do modulo*/
{
...
sys_call_table[SYS_mkdir]=orig_mkdir; /* restaura as chamdas de sistema
originais*/
...
}
```

Quadro 1 - Parte do código de interceptação do sistema.

[1] A função *orig_mkdir* é usada na função *meu_mkdir* - não aparecendo o código completo em razão de espaço.

O Módulo de Monitoramento de Cliente também aciona o módulo de contabilização de Processos (*process accounting*) do Linux, que grava um arquivo de *log* com todos os processos disparados pelo sistema com exceção daqueles que não finalizam corretamente (queda de luz, por exemplo). O quadro a seguir mostra o procedimento que aciona a contabilização de processos do Linux e o conteúdo do arquivo de *log* de contabilização de processos com os registros sobre os processos que foram executados.

```
#include <unistd.h>
...
int Codigo = acct ("/usr/logs/proc.txt");
...
```

Quadro 2 - Parte do código de contabilização dos processos monitorados.

O resultado do trecho de código mostrado acima, que usa a chamada de sistema acct, resulta no arquivo de *log* mostrado na figura a seguir**.**

```
Ac_comm       tty   uid   gid  flag  exit utime stime ctime
Teste        4818     0     0     2     0  0.01  0.00 Wed Oct 10 02:11:32
Ls          34818     0     0     0     0  0.00  0.01 Wed Oct 10 02:11:35
Ps          34818     0     0     0     0  0.01  0.03 Wed Oct 10 02:11:37
Ps          34818     0     0     0     0  0.02  0.02 Wed Oct 10 02:11:39
Ls          34818     0     0     0     0  0.01  0.00 Wed Oct 10 02:11:45
Dir         34818     0     0     0     0  0.00  0.00 Wed Oct 10 02:11:48
Ls          34818     0     0     0     0  0.02  0.01 Wed Oct 10 02:11:51
Ls          34818     0     0     0     0  0.01  0.00 Wed Oct 10 02:11:58
Ls          34818     0     0     0     0  0.02  0.00 Wed Oct 10 02:12:02
ls          34818     0     0     0     0  0.01  0.01 Wed Oct 10 02:12:13
ls          34818     0     0     0     0  0.01  0.01 Wed Oct 10 02:12:19
joe         34818     0     0     0     0  0.04  0.00 Wed Oct 10 02:12:27
ls          34818     0     0     0     0  0.01  0.00 Wed Oct 10 02:12:55
ls          34818     0     0     0     0  0.01  0.01 Wed Oct 10 02:12:57
teste       34818     0     0     2     0  0.00  0.01 Wed Oct 10 02:13:03
teste       34818     0     0     2     0  0.00  0.01 Wed Oct 10 02:13:03
ls          34818     0     0     0     0  0.01  0.01 Wed Oct 10 02:13:06
joe         34818     0     0     0     0  0.01  0.00 Wed Oct 10 02:13:12
shutdown    34818     0     0     3     0  0.00  0.00 Wed Oct 10 02:13:28
...
```

Quadro 3 - Parte do log.

4.6 O Módulo de Diagnóstico (Servidor)

O Módulo de Diagnóstico (Servidor) é responsável pela análise de todos os *log*s gerados pelo sistema. Como já foi dito anteriormente, ele funciona como se fosse um cliente dos Módulos de Monitoramentos de clientes ele, como anteriormente citado, faz requisições para os clientes através do protocolo RPC, que então retornam as informações requisitadas. Este é o único módulo do protótipo com o qual o usuário pode interagir, sendo basicamente sua função retornar relatórios.

4.7 O Módulo de Monitoramento de Rede

O funcionamento do Módulo/Agente de Monitoramento de Rede ocorre da seguinte forma: o módulo roda em um servidor - com a permissão de *root* -, onde ele captura todo o tráfego que circula pela rede; então compara as informações obtidas pela rede com um arquivo de assinaturas – conforme a figura 1 -, a procura de possíveis tentativas de ataques. Caso encontre algum registro anormal, ele então reporta no arquivo de *log* do Módulo de Monitoramento de Rede .

```
FTP-exploit1                  tcp * 21 "|50 57 44 0A 2F 69|"
FTP-exploit2                  tcp * 21 "|58 58 58 58 58 2F|"
# FTP-nopassword              tcp * 21 'pass |0d|'
# FTP-incorrect-login         tcp 21 * "Login incorrect"
# TELNET-Incorrect-login      tcp * 23 "Login incorrect"


SMTP-exploit1                  tcp * 25 'Croot|09090909090909|Mprog,P=/bin'
SMTP-exploit2                  tcp * 25 'rcpt to|3a207c| sed '1,/^$/d'|7c|'
SMTP-exploit3                  tcp * 25 'mail from|3a20227c|'
SMTP-exploit4                  tcp * 25 'Croot|0d0a|Mprog, P=/bin/'
SMTP-exploit6                  tcp * 25 'rcpt to|3a| decode'
SMTP-exploit7(CVE-1999-0204)   tcp * 25 '|0a|C|3a|daemon|0a|R'
SMTP-exploit8(CVE-1999-0204)   tcp * 25 '|0a|Croot|0a|Mprog'
SMTP-exploit9(CVE-1999-0204)   tcp * 25 '|0a|Croot|0d0a|Mprog'
SMTP-exploit10(CVE-1999-0095)  tcp * 25 '|7c 73 65 64 20 2d 65 20 27 31 2c
2f 5e 24 2f 27|'
SMTP-exploit11(CVE-1999-0204)  tcp 113 25 '|0a|D/'
SMTP-expn-decode               tcp * 25 'expn decode'
SMTP-expn-root                 tcp * 25 'expn root'
SMTP-vrfy-decode               tcp * 25 'vrfy decode'
SMTP-Pikachu(Pokey)-Worm       tcp * 25 'pikachupokemon.exe'

...
```

Quadro 4 – Parte do arquivo de assinaturas.

5 Trabalhos futuros e considerações finais

As interfaces dos módulos desenvolvidos estão sendo adaptados para que se possa possibilitar a compatibilidade desejada por IDS de maior porte - como o *Asgaard* [CAM 01], por exemplo. Atualmente, está se verificando o impacto que os mesmos na rede. Da mesma forma, está-se ampliando as possibilidades de diagnóstico. Deseja-se desenvolver determinados módulos, de tal forma que se possam fornecer mesmas funcionalidades de outros programas consagrados - por exemplo, o *snort*.

Uma das alternativas foi a escolha de linguagem de programação que possibilitasse maior portabilidade. Inicialmente, era a intenção o desenvolvimento em Java. Contudo, considerações de projeto incorreram em decidir-se por C++.

Referências Bibliográficas

[CAM 01] CAMPELLO, Rafael; WEBER, Raul; SERAFIM, Vinicius da Silveira; RIBEIRO, Vinicius Gadis. O Sistema de Detecção de Intrusão Asgaard. In: *Workshop* em Segurança de Sistemas Computacionais. WSEG 2001, 2001, Florianópolis. **Anais...** 2001. p. 25-30.

[CER 01] CROSBIE, Mark et al. ***Intrusion Detection Pages***. 2001. Disponível em: <*http://www.cerias.purdue.edu/coast/intrusiondetection/welcome.html*>. Acesso em 21 de março de 2001.

[FIS 01] FISCHER, André; RIBEIRO, Vinicius Gadis. Um Sistema de Detecção de Intrusão projetado para usuário final. In: *Workshop* em Segurança de Sistemas Computacionais. WSEG 2001, 2001, Florianópolis. **Anais ...** 2001. p. 19-24.

[KUM 95] KUMMAR, C. ***Classification and Detection of Computer Intrusions***. 2001. Disponível em <*http://www.researchindex.com*> . Acesso em 23 de março de 2001.

[NED 99] NED, Frank. Ferramentas IDS, 1999. Disponível em <*http://www.revista.unicamp.br/infotec/artigos/frank4.html*>. Acesso em 22 de abril de 2001.