Um Modelo Adaptativo Para Detecção de Comportamento Suspeito em Redes de Computadores

Adriano M. Cansian
(adriano@ibilce.unesp.br)
UNESP - Universidade Estadual Paulista - IBILCE / S.J.Rio Preto
Po Box 136 - CEP 15970-001 - São José do Rio Preto - SP - Brasil

José M. Bonifácio Jr, Edson S. Moreira e André C. P. de L. F. de Carvalho.
(boni, edson, andre@icmsc.sc.usp.br)
Instituto de Ciências Matemáticas de São Carlos - ICMSC/USP
Po Box 668 - CEP 13560-970 - São Carlos - SP - Brasil

Resumo

Todos os sistemas computacionais, notadamente aqueles ligados à Internet, são suscetíveis a ação de atacantes, sejam eles internos ou externos. Neste trabalho apresentamos um modelo de detecção de intrusos em redes de computadores que operam sobre protocolo TCP IP. O sistema consiste do reconhecimento de assinaturas de ataque, baseando-se em captura de pacotes. A identificação do comportamento intrusivo é realizada por intermédio de uma rede neural, o que confere grande adaptabilidade e permite acompanhar as modificações das técnicas de ataque. É descrito um modelo de detecção de intrusos, e apresentada a situação atual de implementação de um protótipo do sistema.

Abstract

All computer systems, mainly those connected to the Internet are vulnerable to attacking attempts, both from internal and external sources. In this work we present a TCP IP based network intrusion detection model. The system consists of a pattern recognizer of attacks, based in capture of packets. The identification of intrusion behavior is realized by a neural network, that provides high adaptability and allows for following up modifications in the attacking techniques. We describe the intrusion detection model and present the current status of the implementation of a prototype.

1. Introdução

O recente aumento no uso de redes de computadores tem sido seguido por preocupações envolvendo segurança e abuso dos recursos computacionais. Além do mais, um crescente número de atividades essenciais são realizadas através de redes de computadores (especialmente sobre a Internet), e portanto uma correta e confiável operação é vital. Por outro lado, atos de pirataria, tentativas de intrusão, invasões consumadas e ações de *break-ins* estão se tornando frequentes e envolvem um alto número de computadores [1-2]. Este cenário cria a necessidade de técnicas especiais de segurança em modernos sistemas de computadores; técnicas que vão além da prática tradicional de se *trancar as portas.*

Este artigo descreve o estado atual do projeto de um sistema de segurança baseado em redes que usa técnicas de filtro e captura de pacotes, e redes neurais para encontrar intrusos, para registrar e aprender sobre seu modo de operação e então fornecer elementos que ajudem o administrador de sistemas a tomar ações contra eles. O sistema de detecção é formado por um ou mais agentes de segurança, que são colocados em pontos estratégicos na rede e que fornecem informação ao pessoal ou sistema de gerenciamento.

2. Detecção de Comportamento Suspeito

Tentativas de ataque acontecem de acordo com algumas técnicas de acesso e frequentemente o invasor está fisicamente fora do sistema sobre ataque [2]. Os primeiros modelos de Sistemas de Detecção de Intrusão (SDI), projetados para computadores isolados, usam algoritmos básicos que incluem análise de funções multinomiais e aproximação de matrizes covariantes para detectar desvio do comportamento normal [3-4], tão bem como sistemas especialistas para detectar violação de políticas de segurança [7]. Os modelos mais modernos monitoram um grande número de redes de computadores e transferem a informação monitorada para ser processada em um equipamento central que emprega técnicas de sistemas distribuídos.

A maioria dos SDIs tem um processo auditor (*daemon*) em cada máquina, responsável por capturar ações de violação de segurança dentro da máquina. Sistemas baseados em redes, ao invés de utilizar pistas de auditoria, analisam o tráfego de pacotes dentro da rede[8] para detectar comportamento intrusivo [5][11].

Uma das propostas de inovação de nosso sistema de gerenciamento de segurança consiste em introduzir um agente de segurança capaz de detectar comportamento intrusivo em conexões estabelecidas.

Este agente atua capturando e decifrando pacotes que são transmitidos através da rede sobre monitoramento. Para fazer uma inferência sobre a condição de segurança das conexões, o agente emprega um sistema especialista e uma rede neural que irá prover um coeficiente de suspeita, o qual, baseado em informações intrusivas previamente registradas, dará uma idéia a respeito da severidade do ataque ou o grau de suspeita das atividades naquela conexão.

O sistema se baseia no fato de que uma intrusão pode ser detectada a partir de uma análise de modelos predeterminados, que são anômalos comparados com ações normais [5-6][11]. A grande maioria dos ataques são resultado de um pequeno número de ataques conhecidos, como relatados por equipes como o CERT [15].

Nossa proposta é que o uso de Redes Neurais pudesse fornecer, com sucesso, mecanismos para o reconhecimento de ataques, tão bem quanto uma capacidade de adaptação em resposta a mudanças nas técnicas de intrusão. Redes neurais artificiais são sistemas distribuídos altamente paralelos compostos por simples unidades de processamento como neurônios, dispostos em uma ou mais camadas. Há um grande número de conexões com pesos entre cada par de neurônios. Estes pesos guardam o conhecimento de uma rede neural e são usados para definir a influência de cada entrada recebida por um neurônio para se produzir uma saída. A saída de um neurônio é frequentemente o resultado de uma função de ativação aplicada a soma ponderada de suas entradas [17-20].

3. O Modelo do Sistema

O agente é colocado em uma máquina segura, que é logicamente invisível às outras. Esta máquina está, ainda, em um lugar onde o acesso físico é restrito, de forma que se possa ter acesso a ela apenas em condições especiais. Adicionalmente, máquinas como essa são colocadas em pontos sensíveis onde possa capturar o tráfego na rede sobre monitoramento. O agente é organizado em um modelo com quatro módulos (figura 1). Os módulos gerenciam o fluxo de pacotes e fornecem um vetor de estímulo para a rede neural. O nível mais baixo apenas captura um fluxo de dados na rede e passa os pacotes ordenados ao segundo módulo. O módulo seguinte consiste de 2 sub-módulos: módulo de pré-seleção de pacotes, e módulo do sistema especialista. O módulo de pré-seleção faz a monitoração e filtragem iniciais dos pacotes, que podem representar eventos de interesse, tais como que tipo de protocolo será monitorado ou que origens e destinos devem ser considerados. Os pacotes previamente filtrados então passam através de uma análise feita pelo sistema especialista. O sistema especialista usa as seguintes informações ao tomar decisões: Quais os caminhos esperados de conexões: origem e destino (quais podem ser perigosos) e portas de origem e destino envolvidas; sensibilidade das máquinas e confiabilidade dos domínios; capacidades dos serviços e autenticação do nível de segurança.

Estes dados funcionam como um elemento de análise para monitorar os eventos de risco, que são armazenados em uma lista de monitoramento de eventos, onde ficam por um tempo definido. Os dados armazenados nesta lista são vetores (figura2 - "Vetores de Monitoramento") que incluem origem e destino da conexão, portas envolvidas, um "nível de segurança" (NS) e um horário. Cada conexão suspeita ou de interesse é representada por um vetor na lista. O "nível de segurança" é um valor numérico que cresce de acordo com a chegada de eventos monitorados vindos de um par origem-destino ou do mesmo domínio. Por exemplo, o sistema detecta um comando *finger* do *host* A para o B, e então inclui este vetor na lista com um nível de segurança previamente estabelecido para o comando *finger* (NS = 10, por exemplo). Quando um novo evento é detectado, por exemplo um *telnet* (NS = 15, por exemplo), a lista é percorrida e uma vez encontrada uma entrada vindo da mesma origem-destino, o nível de segurança deste vetor será acrescido do valor do nível de segurança do *telnet*, sendo agora o novo valor no vetor de conexão NS = 25. Neste caso, não é criado um novo vetor na lista. Outro caso a ser considerado é aquele em que o evento detectado vem de um domínio onde outro evento já aconteceu (já registrado na lista), mas não necessariamente gerado da mesma máquina. Neste caso, um novo vetor é inserido na lista, com o mesmo nível de segurança adicionado de um peso, indicando que uma tentativa prévia vindo daquele domínio já foi detectada.

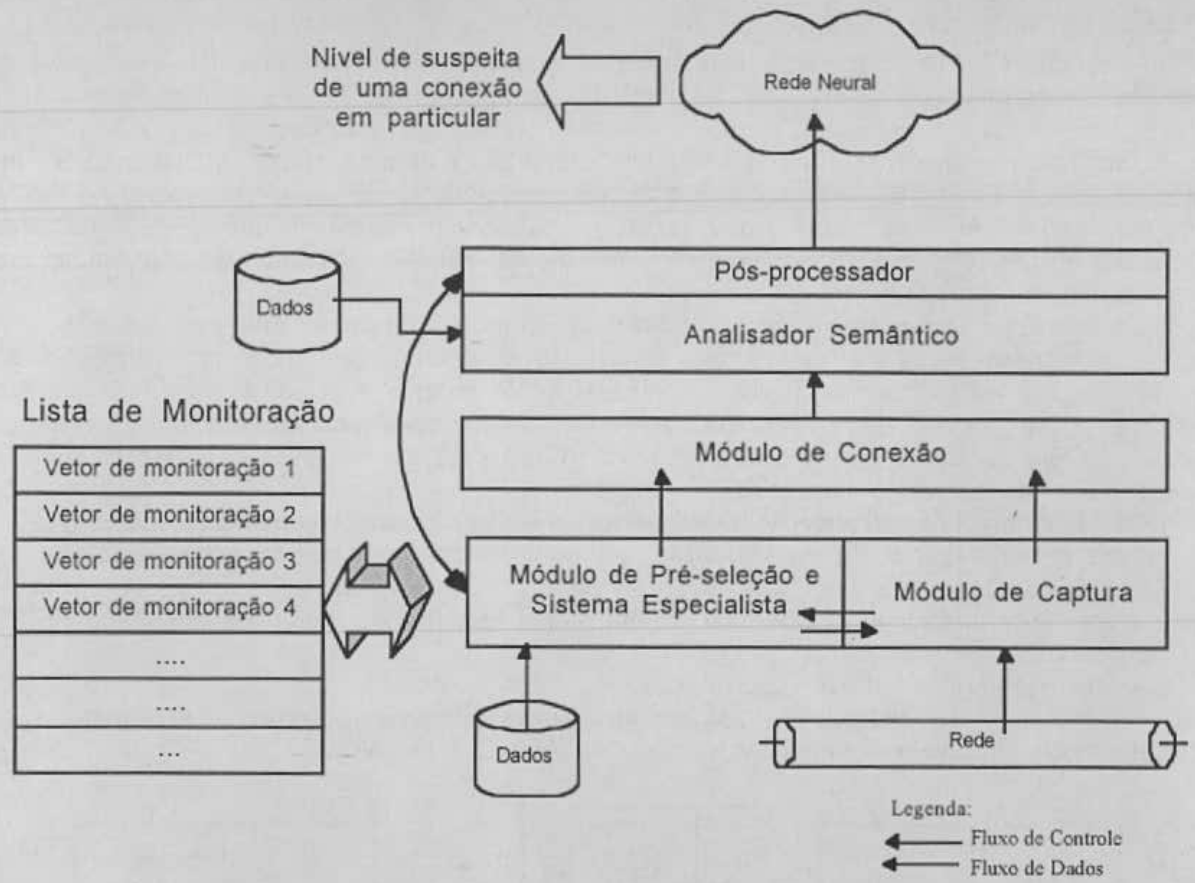

Figura 1 - Estrutura modular do sistema

Esta lista é periodicamente verificada, e quando o nível de segurança de um vetor atinge um limiar pré-estabelecido, o sistema especialista monitora toda a conexão ou, alternativamente, todas as conexões vindas daquele domínio (ainda há a possibilidade de se fazer diferentes tipos de buscas e verificações na lista, considerando não apenas o limiar para um par origem-destino, mas também informações relacionadas sobre os domínios ou um sub-conjunto de endereços relacionados). Desta maneira, quando o sistema especialista identifica padrões de comportamento diferentes dos aceitáveis, todos os pacotes trafegando naquela conexão são mandados para o próximo nível (módulo de conexão). O terceiro módulo é baseado no modelo hierárquico do Monitor de Segurança de Redes (MSR) [8-11]. Ele recebe os pacotes e organiza-os em uma relação de causa-efeito, identificando um fluxo de dados unidirecional. Isto é feito através da análise dos campos de origem e destino, portas, e número de sequência dos pacotes, solucionando então o problema da fragmentação. Este procedimento pode ser feito no nível de IP ou TCP.

Uma vez identificados e ordenados, tem-se isolados os fluxos dos dados sendo transferidos entre máquinas diferentes através de determinadas portas usando determinados protocolos. Estas informações são mapeadas em "Vetores de Fluxo", que são a transcrição dos dados de uma particular conexão. Se necessário, pares de vetores de fluxo são formados para se representar um sequência de dados bidirecional. Os pares de vetores de fluxo são representados por um "Vetor de Conexão", que contém a sequência inteira de dados que está trafegando entre as duas máquinas através da conexão que está sendo monitorada. Os vetores de conexão são então mandados para serem processados pelo "Módulo Reconhecedor de Assinaturas" (Analisador Semântico).

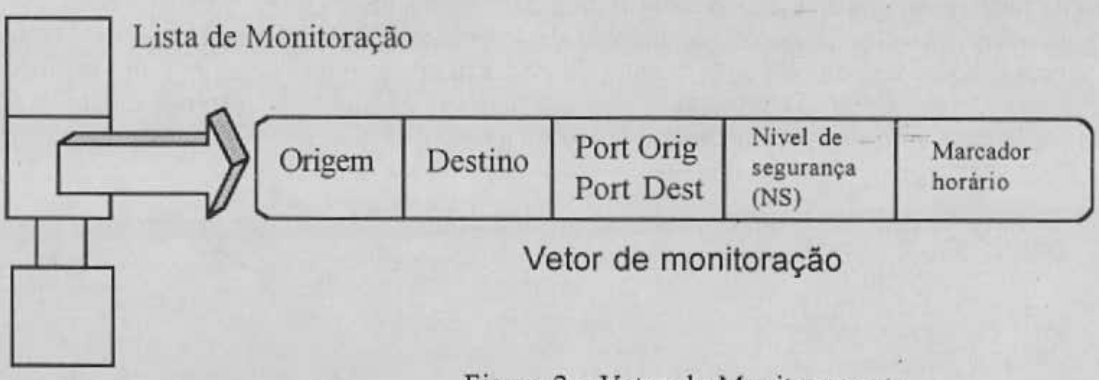

Figura 2 - Vetor de Monitoramento

O Módulo Reconhecedor de Assinaturas atua sobre os vetores de conexão, buscando por perfis de ataque que poderiam aparecer nos dados. Os perfis são guardados em uma base de dados e atualizados de acordo com a necessidade. Estes perfis são assinaturas de ataque e contém informações de como uma sessão suspeita se comporta. Este módulo reconhecedor foi implementado como um autômato finito implementando um analisador léxico/semântico. Então ele pode percorrer o texto correspondente aos dados trafegando na rede e buscar por *strings* especificas (*login root*, *access denied*, etc). Estas *strings* correspondentes às assinaturas de ataque ou *strings* de controle (tipo e portas da conexão, direção dos pacotes, etc) que são convertidas em códigos binários para entrada na rede neural.

Estes códigos são agrupados por conexão e usados para formar o vetor de estímulo para a rede neural. Estas informações são mandadas para o último módulo de pós-processamento, que unifica as informações do módulo de reconhecimento de assinaturas (perfis associados) às do sistema especialista (sensibilidade dos serviços e conexões) e à sua base de dados, para formar o vetor de estímulo para a rede neural. O vetor de estímulo (figura 3) contém todos os valores importantes e informações consideradas importantes sobre a conexão, em um formato binário. Estes valores são:

- Capacidade do serviço e nível de autenticação: *telnet* oferece mais capacidades e autenticação que SMTP;
- Nível de segurança das máquinas de origem e destino, que pode ser um valor numérico fornecido por um sistema analisador de padrões, como o SATAN ou Tigger Scripts, ou mesmo um valor padrão ou um valor que referencia a confiança do sistema operacional, atribuídas por organizações internacionais como FIRST [14], CERT [15] ou COAST [16];
- Quantidade de dados transferidos e horário da conexão;
- E, finalmente, a sequência dos códigos binários das *strings* suspeitas, componentes das assinaturas de ataque, localizados pelo analisador semântico nos dados, tal como descrito acima.

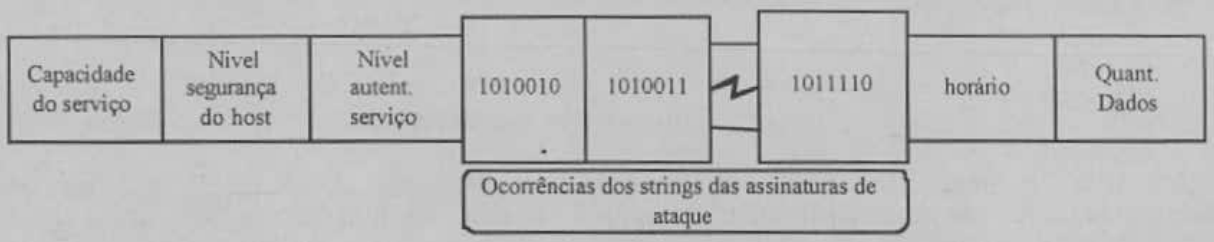

Figura 3 - Vetor de Estímulo

A rede neural analisa o estímulo, e tenta atribuir um grau de suspeita, que representa o estado de suspeita de uma conexão em particular. Antes que a rede neural possa identificar ataques potenciais, ela deve ser treinada com um significativo e suficientemente grande número de vetores de estímulo, que representem o comportamento de conexões suspeitas e legítimas. Uma vez treinada, a rede deve usar suas características de generalização para identificar corretamente os usuários que mostrem características similares às contidas nas ações de intrusão usadas para treiná-la.

O sistema de gerenciamento (remoto) recebe então o valor numérico que representa o estado de segurança de conexões suspeitas e mapeia-os, classificando por grau de segurança, ajudado por um código de cores. Deste ponto, muitas operações podem ser feitas, como a transmissão de diferentes níveis de alerta ou perigo aos administradores (dependendo do nível de risco nos eventos detectados), disparar processos de *log*, ativação de contra-medidas para isolar o *host* ou o domínio que causou o ataque (ativar ajustes de *wrappers* [12], *firewalls* ou filtros [13]), entre outros. A rede neural deve também ser ajustada, para ser capaz de refletir mudanças no padrão de comportamentos intrusivos. O administrador pode detectar comportamentos altamente intrusivos, mas descobrir que a rede neural não está identificando corretamente ou não está identificando sua severidade como o esperado. Neste caso, novos padrões podem ser adicionados à rede e ela ser retreinada para que aprenda a reconhecer novos padrões adequadamente. Esta capacidade adaptativa é a mais importante inovação neste projeto e não aparece em outros SDI conhecidos.

O módulo de gerenciamento é independente do módulo de segurança. Os dois módulos se comunicam via TCP/IP que abre a possibilidade de se colocar o módulo de gerência em qualquer lugar da rede, inclusive para que um agente possa servir a uma comunidade de gerentes. O módulo de gerenciamento será desenvolvido como uma aplicação num sistema qualquer de gerenciamento de redes.

4. Implementação de um Protótipo do Sistema

4.1 Introdução

A fim de testar a funcionalidade de nosso modelo proposto de detecção de intrusos, procedemos ao desenvolvimento de um protótipo. Este protótipo possui algumas simplificações com relação ao modelo proposto, principalmente com relação ao módulo de pré-seleção de conexão, onde o sistema especialista foi inicialmente aproximado e substituído por um conjunto simples de regras. Além disso, algumas informações não foram incluídas no vetor de estímulo da rede neural, como por exemplo o nível de segurança das máquinas envolvidas, mencionado anteriormente. Este tratamento foi inicialmente incorporado ao módulo de pré-seleção, de forma que o vetor de estímulo da rede neural possui essencialmente a porta utilizada na conexão mais as possíveis ocorrências das *strings* suspeitas, obtidas pelo analisador semântico e convertidas em binários.

Optou-se por implantar todos os módulos do protótipo, em uma única máquina, a fim de eliminar a necessidade de desenvolvimento de agentes e gerentes de comunicação de dados. O ambiente de abrangência considerado trata-se do barramento de uma subrede *ethernet* (figura 4), com tráfego de pacotes TCP/IP.

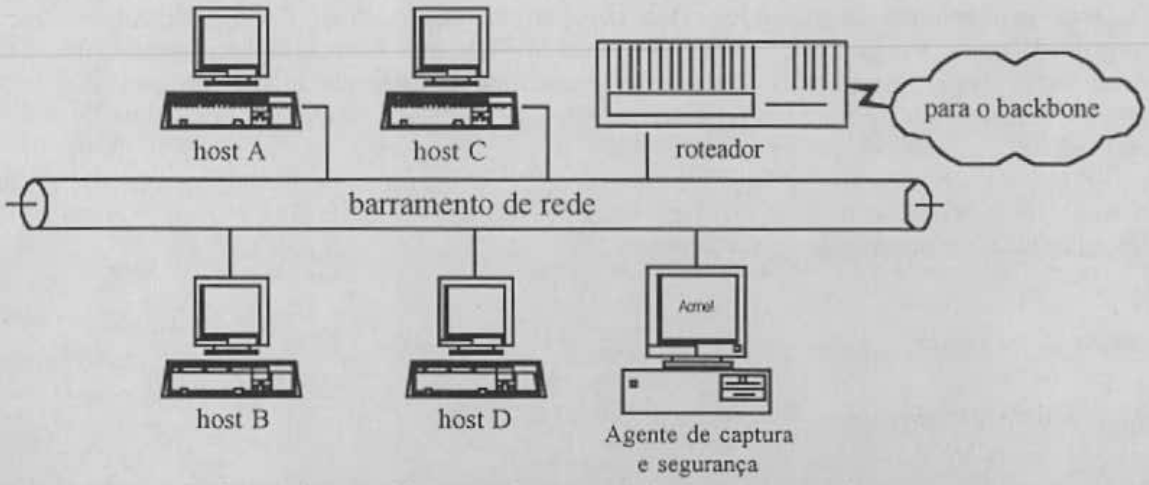

Figura 4 - Ambiente onde o protótipo realiza monitoração

4.2 O módulo de captura de pacotes

4.2.1 Características gerais

A captura de pacotes é realizada por um módulo que se encontra entre o módulo de pré-seleção, que analisa os dados enviados, e o meio físico por onde trafegam todas as informações da rede. Este módulo é controlado pelo módulo de pré-seleção. Através de uma biblioteca de captura de pacotes, o módulo tem acesso à interface de rede e aos pacotes que passam por ela, mapeando-os para uma estrutura em memória que segue o padrão de um datagrama IP. O módulo realiza três operações básicas: a captura preliminar pacotes, análise inicial dos dados e captura intensiva dos pacotes.

4.2.2 A biblioteca de captura de pacotes

Para a captura de pacotes foi utilizada a biblioteca *libpcap v0.06*, que na verdade trata-se de uma API (*Application Program Interface*) de implementação do *BSD Packet Filter* (BPF) [25], o que a torna rápida, eficiente, e de simples utilização. Esta API permite a captura não apenas de pacotes *ethernet*, mas também de pacotes PPP e SLIP, bastando para isso definições de código apropriadas.

O modo de operação normal do dispositivo de rede não permite o acesso a pacotes que não sejam destinados à máquina onde o sistema está executando. No entanto, existe um modo de operação, chamado de modo promíscuo, que permite que todos os pacotes sejam acessíveis. A execução de controle da interface de rede em modo promíscuo é reservada a processos que sejam executados com prioridade de *root* (UID zero).

Dessa maneira, executadas com prioridades apropriadas, as funções da biblioteca colocam o dispositivo de rede em modo promíscuo e então recebem uma cópia de cada pacote que passa por ele. Tanto pacotes de entrada como de saída são acessíveis. A partir daí os pacotes são filtrados apropriadamente conforme a solicitação, e assim, o processo que fez a chamada recebe apenas os dados que foram solicitados.

Esta filtragem é realizada a nível de *kernel* da máquina. Vários processos podem ser executados, todos utilizando a *libpcap* e especificando uma filtragem diferente no mesmo dispositivo de rede.

Além de especificar os parâmetros da filtragem desejada, discutidos a seguir, o processo originador da chamada deve especificar um valor de *timeout* para a biblioteca. Como normalmente o tráfego de uma rede é intenso, ao invés de passar os pacotes um a um para o processo, a biblioteca tenta utilizar um *buffer* e retornar somente quando este está cheio ou o *timeout* definido pelo processo expirou.

4.3 O módulo de pré-seleção

A fim de simplificar alguns procedimentos de implantação, o módulo de pré-seleção de conexão não incorporou o sistema especialista, inicialmente proposto no nosso modelo. Um pequeno conjunto de regras simples, que em síntese decide quais tipos de pacotes e conexões são inicialmente monitorados, substituiu o sistema especialista. Entretanto esta generalização não implica em nenhuma perda significativa em nosso modelo, nem na identificação do comportamento intrusivo. Inicialmente optamos por monitorar apenas pacotes provenientes de redes externas, ou seja, não pertencentes à rede local sob monitoramento.

Assim, o módulo de pré-seleção faz as chamadas apropriadas ao módulo de captura, solicitando que apenas os pacotes provenientes de sub-redes externas sejam analisados. Além disso, determina-se que a análise somente se proceda quando o pacote contém dados do tipo TCP, que correspondem aos eventos de interesse (FTP, *telnet*, http, *finger*, *ping*, etc). Para este procedimento, apenas alguns campos dos pacotes são preliminarmente examinados de modo a verificar se satisfazem essas características iniciais de pré-filtragem. Esses campos são:

- No cabeçalho IP: o endereço IP de origem e o endereço IP de destino; e
- No cabeçalho TCP: a porta origem e a porta destino.

4.4 O módulo de conexão

4.4.1 Características gerais

Quando o módulo de pré-seleção decide pela monitoração completa (*headers* e campos de dados) de uma conexão, o módulo de captura passa a enviar os dados dos pacotes monitorados para o módulo de conexão. O módulo de conexão utiliza as mesmas estruturas e funções do módulo de captura, e passa a criar os vetores de conexão, que se tratam dos mapeamentos dos dados bidirecionais de uma conexão em particular. O módulo de conexão foi inicialmente implementado em conjunto com o módulo de pré-seleção e captura. Depois foi transformado em um processo independente dentro do sistema. É importante ressaltar que, neste trabalho, o termo "conexão" indica apenas um conjunto de pacotes de informações cujos endereços origem-destino são os mesmos, e não está relacionado com denominação de "orientado a conexão" e "sem conexão", como normalmente presentes na literatura.

4.4.2 Mapeamento dos dados

O vetor de conexão é copiado em uma unidade de saída. Antes do mapeamento dos dados, o módulo de conexão gera um registro que contém, nesta ordem, tipo dos pacotes da conexão (se TCP, UDP ou ICMP), endereço IP da máquina origem, seguido da porta da máquina origem, e endereço IP da máquina destino, seguido da porta de destino. Estas informações são precedidas e finalizadas por alguns caracteres especiais (no caso estes caracteres são o *string* ::::::::::::::), que serão utilizados pelo módulo analisador semântico. O uso desses caracteres será discutido mais adiante. Dessa forma, o identificador da conexão é escrito da seguinte maneira:

```
::::::::::::::
tipo:IPOrigem-PortaOrigem_IPDestino-PortaDestino
::::::::::::::
```

Por exemplo, considerando-se uma conexão TCP da porta 1899 da máquina 143.107.228.1 (origem) para a máquina 200.17.28.8 (destino) na porta 21 (uma conexão FTP), teríamos um registro idendificador de conexão inicial, da seguinte forma:

```
::::::::::::::
TCP:143.107.228.1-1899_200.17.28.8-21
::::::::::::::
```

Em seguida os dados da conexão em ambos os sentidos são copiados. A origem e o destino de cada fluxo de dados é demarcada com dois pequenos grupos de caracteres especiais. Estes pequenos conjuntos de caracteres, que garantidamente não fazem parte de assinaturas de ataque, também serão utilizados pelo analisador semântico. São eles :

- **ACME)** : indica que os dados que o seguem são provenientes da origem. Seguindo o exemplo acima, estes dados viriam de 143.107.228.1
- **ACME(** : indica que os dados que o seguem são provenientes do destino da conexão, no exemplo, vindos de 200.17.28.8

Entretanto, a ordem dos endereços origem e destino, no início do registro, pode estar invertida, dependendo da situação da conexão no momento em que o módulo de pré-seleção disparou a captura completa. Genericamente, o marcador **ACME)** indica o fluxo de dados proveniente da máquina que aparece inicialmente no registro identificador de conexão e **ACME(** indica o fluxo de dados do endereço que aparece em seguida.

```
::::::::::::::
TCP:143.107.228.1-1899_200.17.28.8-21
::::::::::::::
ACME(220 wolverine FTP server ready.
ACME)SITE exec vulnerable/ftp.
ACME(500 'SITE EXEC vulnerable/ftp': command not understood..
ACME)USER anonymous.
ACME(331 Guest login ok, send e-mail address as password.
ACME(530 Please login with USER and PASS..
ACME)PASS -intruso@algum.lugar.br.
ACME(230 Guest login ok, access restrictions apply..
ACME)PWD.
ACME(257 "/" is current directory..
ACME)MKD teste
ACME(550 teste: Permission denied..
ACME)RMD teste
ACME(550 teste: No such file or directory..
ACME)LIST -Rlag.
ACME(150 Opening ASCII mode data connection for /bin/ls..
ACME(226 Transfer complete..
ACME)QUIT.
ACME(221 Goodbye..
```

Este registro é, a grosso modo, uma assinatura de ataque, que contém todo o desenrolar do comportamento de uma conexão suspeita. Entretanto, como há muitas informações desnecessárias, é necessário proceder a uma redução desses dados, passando-os para o analisador semântico. É o analisador semântico que irá filtra-los, buscando pelos componentes principais das assinaturas de ataque, para finalmente produzir um vetor binário que traduz este comportamento suspeito, que será interpretado pela rede neural.

4.5 O analisador semântico e pós-processador

4.5.1 Características gerais

O analisador semântico trata-se de um sistema de reconhecimento e codificação de *strings* suspeitas. A função básica deste módulo é reconhecer componentes das assinaturas de ataque pré-estabelecidas, encontradas numa porção de texto, bem como identificar: que tipo de conexão se trata (TCP, UDP, ICMP), as portas envolvidas; o início e fim da conexão; e em que sentido está indo a *string* reconhecida da assinatura com relação ao fluxo de dados (se do *host A* para o *host B* ou vice-versa). A entrada deste módulo é um registro de conexão, gerado pelo módulo de conexão conforme discutido anteriormente. O analisador interpretará os grupos de caracteres de controle utilizados e os dados trafegados.

A codificação das assinaturas em binários é feita através de um código decimal intermediário. Este código é um número inteiro positivo menor que 30000. Valores acima estão reservados para reconhecer inícios e finais de conexão, tipo de conexão, dentre outras operações de controle. Existe então uma segunda tabela que mapeia estes códigos inteiros em códigos binários. Todo este processo é feito visando uma maior simplicidade e independência das seções do módulo, facilitando inserções de novas assinaturas. Seguindo este

raciocínio, é possível haver vários códigos inteiros que codifiquem *strings* diferentes mas que tenham o mesmo significado (*login ftp* e *login guest*).

O arquivo de dados que contém os *strings* das assinaturas (tsr.acm) é composto da seguinte forma: a primeira linha contém um inteiro indicando quantos registros existem, e depois linha por linha, um inteiro, que é o código, seguido da *string* da assinatura, podendo mais de uma *string* compartilhar o mesmo código inteiro. Chamaremos este arquivo de Tabela de Símbolos Reservados (TSR). Há ainda outro arquivo que codifica inteiros em binários (bin.acm), e contém um código inteiro o seu respectivo código binário. Esta codificação em dois passos visa otimizar memória e redundância, uma vez que há casos em que mais de cem strings compartilham o mesmo código binário, isso evita que se tenha esse mesmo código replicado várias vezes e garante a integridade os dados. Estes arquivos são lidos sempre que o módulo é carregado.

As *strings* das assinaturas, e seus respectivos códigos inteiros, são lidos e armazenadas numa tabela *hash* e os binários em um vetor. A tabela *hash* visa otimizar a velocidade nas buscas pelas *strings* nas comparações com o texto de entrada. Como após construída a tabela não mais se precisará fazer inserções ou remoções (complicada em tabelas *hash*), a tabela *hash* apresenta uma solução melhor que busca binária em vetores ou estruturas como árvores de busca binária. Os códigos binários, como são poucos, são mantidos em um pequeno vetor onde o índice é o próprio código inteiro associado a ele. Isto reduz espaço, além do que torna rápida a busca por um código binário.

Todas estas preocupações com relação à memória e desempenho são justificadas uma vez que este processo deve ser feito em tempo real, podendo-se ter uma grande quantidade de dados em casos extremos. Além disso, deve-se contar ainda com o tempo de processamento da rede neural.

4.5.2 Esquemas de codificação das assinaturas para código binário

A codificação dos conjuntos de caracteres suspeitos das assinaturas em códigos binários merece especial atenção. A forma de codificação é uma importante fase no desenvolvimento e treinamento de redes neurais. Uma codificação bem estruturada pode levar a um ganho de rendimento tanto no treinamento, quanto na proporção de acertos que a rede apresentará. Uma codificação mal escolhida pode levar a rede a confundir padrões diferentes entre si, o que acarreta em um maior tempo de treinamento para que a rede consiga identificar os padrões corretamente. Além disso, afetará na taxa de acertos que a rede apresenta, uma vez que sua capacidade de abstração pode ser prejudicada ao ser exposta a um novo padrão, tendo de classificá-lo em uma das classes treinadas.

O esquema de codificação utilizado trabalha com códigos de 11 bits, e é feito de forma que cada código difira de todos os outros em pelo menos 4 dos bits, ou seja, garante-se uma certa diferença, ou distância qualitativa, entre quaisquer dois códigos. Isso é feito para que a rede consiga diferenciar bem um código de outro, uma vez que se tivéssemos códigos diferindo em apenas um bit, eles seriam muito parecidos para a rede neural.

Nesse esquema de codificação tendo-se pelo menos 4 bits diferentes, foram conseguidos uma média de 49 códigos diferentes, dos quais 41 usados. A atribuição dos códigos binários é feita de forma aleatória sendo que *strings* que tenham o mesmo significado terão códigos inteiros iguais, e consequentemente, binários também iguais, reduzindo o número de códigos diferentes.

4.5.3 Vetor de Estímulo

O vetor de estímulo é composto por duas partes. Sendo a primeira a porta de destino da conexão.

As portas são codificadas em binário da seguinte forma: como o *telnet* tem uma maior capacidade que *ftp*, que por sua vez tem mais que *SMTP*, e sendo estes os serviços mais frequentes, cada um recebeu um código de 4 bits único e um quarto código comum para as demais portas. Os códigos são 1000 para *telnet*, 0100 para *ftp*, 0010 para *SMTP* e 0001 para as demais portas, além do que, conexões que utilizem outros protocolos como ICMP, que não utilizam portas, recebem o valor 0000. Foi tomado cuidado para que cada código fosse completamente diferente do outro (uma vez que com 2 bits já seria possível 4 combinações diferentes) para que a rede pudesse identificar mais facilmente cada uma das portas. Esta identificação também é importante à medida que cada comportamento intrusivo está ligado diretamente à porta utilizada, uma vez que as *strings* suspeitas (geralmente comandos) são dependentes do tipo que serviço que se está usando. Para fazer com que a rede reconheça que o código 1000 que identifica um *telnet* é mais perigoso que 0100 (*ftp*) de forma que a rede aprenda que 1000 tem mais peso que 0100, coloca-se mais exemplos de ataques utilizando *telnet* que de *ftp* e assim por diante. Com isso, só o fato de ser uma conexão *telnet* já confere à ela um nível maior de perigo que uma conexão *ftp* para a rede neural.

A segunda parte do vetor de estímulo é formada por dez slots de tamanho fixo (11 bits, que é o tamanho dos códigos binários mencionados anteriormente) onde serão inseridos os códigos das *strings* suspeitas na ordem em que aparecem. Estes códigos estão ordenados de acordo com sua ocorrência dentro da

conexão. Esta ordenação é importante uma vez que o comportamento intrusivo é fortemente caracterizado pela sequência em que as *strings* suspeitas aparecem. Por exemplo: um *Login root* seguido de *login incorrect* ou um *mkd* seguido de *permission denied*. Caso se encontre menos *strings* na conexão, os slots restantes são preenchidos com um código nulo (uma sequência de 11 zeros). Desta forma, o vetor de estímulo terá 114 bits. Com isso, uma conexão normal terá um número grande de zeros uma vez que serão encontrados poucas *strings* suspeitas e a maioria dos slots será preenchida com o código nulo.

Por simplicidade, optamos-se por utilizar um simulador de redes neurais para testar nosso modelo. O simulador escolhido trata-se do SNNS (*Stuttgart Neural Network Simulator)* [23]. Este simulador realiza as fases de treinamento da rede, além de gerar um código C da rede treinada que pode ser facilmente incorporado ao sistema. A rede neural utilizada é uma rede *feed-forward* com treinamento *back-propagation*. Esta rede tem 114 neurônios de entrada correspondentes aos 114 bits do vetor de estímulo e um neurônio de saída, o qual fornece o grau de suspeita que a rede atribuirá a esta conexão. A camada intermediária será testada com diferentes números de neurônios para avaliar performance e grau de acerto.

5. Resultados obtidos e situação atual

Atualmente, o Módulo de Captura foi testado com sucesso, e diversas tentativas de intrusão foram monitoradas. O analisador semântico foi implementado com sucesso e cerca de 120 tipos diferentes de ataques foram simulados e capturados em um ambiente de rede controlado. Essas simulações foram realizadas manualmente através de técnicas de intrusão conhecidas, e através dos sistemas SATAN [24] e ISS [22]. Isso permitiu obter um bom conjunto de assinaturas de ataque para treinar a rede neural. No momento, estas assinaturas estão sendo utilizadas no simulador de rede neural e os resultados estão sendo analisados.

6. Conclusões

Nosso Sistema de Detecção de Intrusão pretende prover uma rápida e versátil ferramenta para abordar o problema de intrusão em redes de computadores. Ele se baseia no fato de que a maioria dos atos de intrusão seguem determinados padrões. Embora ainda não tenhamos resultados suficientes para demonstrar a acuracidade do método, acreditamos que existe uma grande chance de sucesso. Tal sucesso é reforçado por soluções similares, como determinação do comportamento no uso de cartões de crédito [21] (se é um uso normal ou indevido) através de redes neurais. Este método não tenta dar uma decisão definitiva sobre a presença de uma intrusão, mas sim, fornecer um nível que indique o grau de atividades intrusivas.

Outra importante característica deste método é a adaptabilidade. O conhecimento dos administradores é facilmente introduzido no sistema (e na rede neural retreinada) de forma que novas informações importantes no processo podem ser incorporadas dinamicamente, tornando o sistema sempre atualizado com novas técnicas de intrusão.

7. Agradecimentos

Os autores agradecem o apoio da FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo e da CAPES - Coordenadoria de Aperfeiçoamento de Pessoal de Ensino Superior, no financiamento deste projeto.

8. Referências

[1] R. Bace. *A New Look at Perpetrators of Computer Crime*. In Proc. 16th Department of Energy Computer Security Conference, 1994.

[2] P. Neumann & D. Parker. *A Summary of Computer Misuse Techniques*. In Proc. 12th National Computer Security Conference, pages 396-407, 1989.

[3] H.S.Javitz & A.Valdez. *The SRI IDES Statistical Anomaly Detector*. In Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1991.

[4] J.R.Winkler & W.J.Page. *Intrusion and Anomaly Detection in Trusted Systems*. In Proc. Fifth Annual Computer Security Applications Conference, Tucson, AZ, Dec,1990. 115-124.

[5] D.E. Denning. *An Intrusion-Detection Model*. IEEE Trans. on Software Engg. Vol. SE-13, pp. 222-232, Feb.1987.

[6] T.F. Lunt et al. *IDES: A Progress Report*. In Proc. Sixth Annual Computer Security Applications Conference. Tuscon, AZ, Dec. 1990.
[7] T.F. Lunt et al. *A Real Time Intrusion Detection Expert System (IDES)*. Interim Progress Report, Project 6784, SRI International, May 1990.
[8] L.T. Heberlein et al. *A Network Security Monitor*. In Proc.1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1990, pp 296-304.
[9] L.T. Heberlein et al.. *Towards Detecting Intrusions in a Networked Environment*. In Proc. 14th DOE Conference on Computer Security. Concord, CA, May 1991, pp 17-47.
[10] L.T. Heberlein, K.N. Levitt and B. Mukherjee. *A Method to Detect Intrusive Activity in a Networked Environment*. In Proc. 14th National Computer Security Conference. Washington, DC. Oct. 1991, pp 362-371.
[11] L.T. Heberlein, B. Mukherjee and K.N. Levitt. *Internet Security Monitor: An Intrusion Detection System for Large-Scale Networks*. In Proc. 15th National Computer Security Conference. Baltimore, MD. Oct. 1992. and B. Mukherjee , L.T. Heberlein and K.N. Levitt. *Network Intrusion Detection*. IEEE Network may/june 1994., pg. 26-41.
[12]W.Venema. *TCP Wrapper: Networking Monitoring, Access Control and Booby Traps*. ftp://cert.sei.cmu.edu/pub/network_tools/tcp_wrapper.txt
[13] S.Garfinkel & G.Spafford. *Pratical UNIX and Internet Security*. O'Reilly and Associates, 2nd Edition, 1996.
[14] http://www.first.org/first/
[15] http://www.sei.cmu.edu/technology/cert.cc.html
[16] http://www.cs.purdue.edu/coast/coast.html
[17] Rumelhart, D.E.; McClelland, J.L. and The PDP Research Group. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, MIT Press 1986.
[18] Carvalho, A.C.; Fairhurst, M.C. e Bisset, D.L. *An integrated Boolean neural network for pattern classification*, Pattern Recognition Letters, Vol. 15, pags 807-813, august 1994.
[19] Lupo, C.J. *Defense Applications of Neural Networks*, IEEE Communications Magazine, Vol. 27, No. 11, pag. 82-88, november 1989.
[20] Wong A. J. *Recognition of General Patterns Using Neural Networks*, Biological Cybernetics, Springer-Verlag, Vol. 58, pag. 361-372, 1998.
[21] Reategui, E. B.; Campbell, J. *A Classification System for Credit Card Transactions*, Department of Computer Science, University College London.
[22] http://iss.net/
[23] http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html
[24] http://flying.fish.com/satan/
[25] S. McCanne and V. Jacobson. *The BSD Packet Filter: A New Architecture for User-level Packet Capture*. In Proc. of the 1993 Winter USENIX Technical Conference, San Diego, CA, USA. January 1993.