Um sistema de captura de pacotes para uso em segurança de redes

Adriano M. Cansian, Aleck Zander T. de Sousa e Sérgio Antônio Leugi Filho
UNESP - Universidade Estadual Paulista
Instituto de Biociências, Letras e Ciências Exatas de S.J.Rio Preto
Po Box 136 - CEP 15970-001 - São José do Rio Preto - SP - Brazil
(E-mail: *adriano@nimitz.dcce.ibilce.unesp.br, aleck,leugi@ensino.ibilce.unesp.br)*

Edson S. Moreira
USP - Universidade de São Paulo
Instituto de Ciências Matemáticas de São Carlos
Po Box 668 - CEP 13560-970 - São Carlos - SP - Brazil
(E-mail: *edson@icmsc.sc.usp.br*)

Resumo

Com o rápido crescimento da Internet tanto em termos do número de máquinas conectadas quanto ao número e tipo de serviços oferecidos, as preocupações envolvendo segurança têm se tornado um tema chave para os projetistas destas tecnologias. Neste trabalho apresentamos a situação atual de desenvolvimento de um sistema de captura de pacotes TCP/IP utilizado para a obtenção de assinaturas de ataque na determinação de comportamento anômalo para detecção de intrusos em redes de computadores.

Abstract

As the Internet expands both in number of hosts connected and in terms of the number of services provided, worries concerning security has become a key issue for the technology developers. In this paper we show the present development status of a TCP/IP packet capture system being applied to obtain attack signatures and to detect intruders into computer networks.

1. Introdução

A demanda por redes de computadores interligadas através de Intranets de todos os portes, traz consigo responsabilidades com a segurança dos dados trafegados e armazenados. Existe uma grande preocupação com o funcionamento correto e confiável destas redes, haja visto que cada vez mais delas dependem atividades essenciais, de todos os tipos. Além disso a Internet tem experimentado um crescimento muito significativo de atividades comerciais sendo desenvolvidas pelo seu intermédio. Por outro lado, os ataques intrusivos a redes e computadores tem crescido tanto em número, quanto em quantidade de máquinas envolvidas [Bac94] [NP89]. Isso faz com que técnicas especiais de segurança se tornem indispensáveis nos sistemas computacionais modernos.

Sabe-se também que, com o passar dos anos e com a disseminação de informações por diversos meios, notadamente através da Internet, houve um crescimento significativo no entendimento de como os sistemas e redes operam, fazendo com que intrusos se tornassem especialistas em determinar e explorar fraquezas de modo a obter acesso privilegiado. Estes mesmos intrusos também passaram a usar modelos de intrusão que são difíceis de traçar e identificar. Eles freqüentemente usam diversos níveis de ocultação e dissimulação antes de realizarem o ataque, procurando despistar o alvo e raramente são surpreendidos realizando ataques massivos repentinos que evidenciem facilmente uma atividade suspeita ou anormal. Eles

também costumam cobrir seus rastros, de forma que as atividades realizadas no sistema penetrado não são facilmente descobertas.

Desse modo, métodos eficazes para a proteção e prevenção de ataques em computadores tornam-se cada vez mais necessários. Sistemas de detecção de intrusão assumem este papel de última linha de defesa dentro do esquema de proteção de computadores. Eles são úteis não somente para se detectar falhas de segurança que foram ou podem ser exploradas com sucesso, mas também para monitorar tentativas de invasão na segurança, o que fornece mecanismos importantes para a adoção de contramedidas, dentro de um tempo apropriado.

Nesse contexto apresentamos um sistema de captura de pacotes para ser utilizado em sistemas de detecção de intrusos em redes, através da obtenção de assinaturas de ataque, como proposto em [CMM+96] e [CMC+96]. O sistema de captura de pacotes foi desenvolvido em ambiente UNIX e é baseado no protocolo de comunicação TCP/IP, que é sabidamente muito utilizado para a interconexão de computadores de várias arquiteturas diferentes.

Através do monitoramento da rede, o sistema captura os pacotes em tráfego, filtra os dados relevantes e os armazena, para que possam ser analisados, e assim se determinar se está ou não havendo uma tentativa de intrusão na rede sob monitoramento.

2. Aplicação da biblioteca de captura de pacotes: *BSD Packet Filtering*

Muitas versões do Unix possuem ferramentas para captura de pacotes, tornando possível o uso de estações de trabalho para o monitoramento da rede. Como esses monitores são executados no nível de usuário, os pacotes devem ser copiados do núcleo do sistema operacional (*kernel*) para o nível do usuário antes de poderem ser analisados. Este processo de cópia pode ser minimizado desenvolvendo um agente no *kernel* chamado de filtro de pacotes (*packet filter*), que descarta os pacotes desnecessários o mais rápido possível. O *packet filter* original do Unix havia sido desenvolvido sobre um filtro baseado em pilhas que tinha uma baixa performance nos atuais processadores RISC. O BPF - BSD *Packet Filter* [MCV92] usa uma nova abordagem, com um filtro baseado em registradores que é 20 vezes mais rápido do que o sistema original. O BPF também usa uma estratégia de *straightforward buffering* que torna sua performance até 100 vezes mais rápida do que o NIT [Sun90] da Sun executando na mesma máquina.

O BPF possui dois componentes principais: o interceptador de rede (*network tap*) e o filtro de pacotes. O interceptador de rede coleta cópias dos pacotes do dispositivo de rede e as entrega às aplicações que estão monitorando a rede. O filtro decide se o pacote será aceito e em caso positivo, quanto do mesmo será copiado para a aplicação.

Quando um pacote chega na interface de rede, o *link-level device driver* envia o mesmo para uma pilha de protocolo do sistema. Mas quando o BPF está "escutando" esta interface, o *driver* primeiro chama o BPF. Este por sua vez envia o pacote para cada filtro associado com cada processo monitorando a rede. Cada filtro decide se o pacote será aceito e quantos bytes deverão ser lidos. Para cada filtro que aceita o pacote, o BPF copia a quantidade de dados pedida para o *buffer* associado (figura 1). O controle então é devolvido ao dispositivo de rede e os procedimentos normais do protocolo de comunicação seguem adiante.

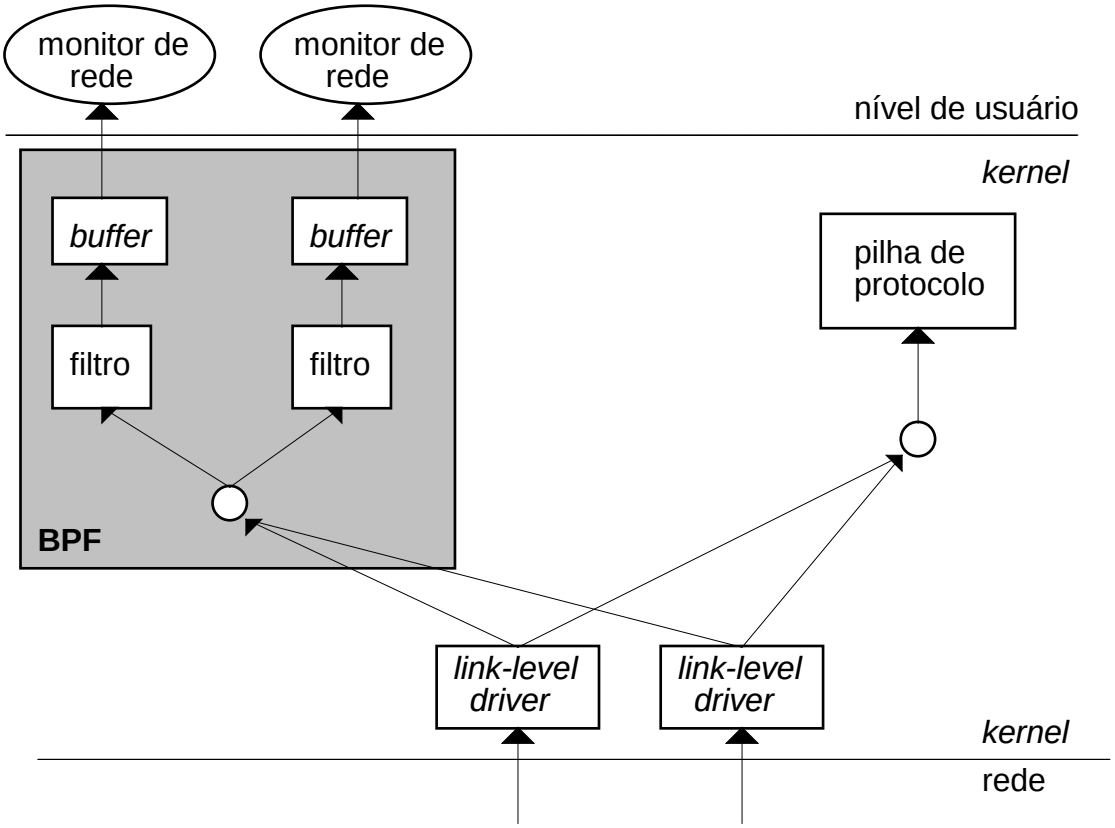

Figura 1 - Funcionamento do BPF.

Como normalmente os processos de monitoramento devem verificar cada pacote na rede e o tempo entre os pacotes pode ser de apenas alguns microssegundos, não é possível fazer uma chamada de leitura do sistema por pacote. Assim o BPF coleta os dados de vários pacotes e os retorna como uma única unidade, quando o processo de monitoramento faz a leitura, encapsulando os dados capturados de cada pacote com um cabeçalho que inclui um *time stamp*, comprimento e deslocamento para o alinhamento dos dados.

Pelo fato de que os processos de monitoramento precisam de apenas um pequeno subconjunto do tráfego da rede, um grande aumento de performance é conseguido filtrando os pacotes indesejados no lugar onde o DMA (*Direct Memory Access*) do dispositivo de rede os coloca ao invés de copiá-los para algum outro *buffer* do *kernel* e filtrá-lo depois. Então se o pacote não é aceito, somente os bytes necessários para processo de filtragem são referenciados.

No sistema em discussão foi utilizada a biblioteca *libpcap v0.3* para a captura de pacotes, que é uma implementação do BPF sendo assim rápida, eficiente e de simples utilização. Esta biblioteca permite a captura não apenas de pacotes *Ethernet*, mas também de pacotes PPP (*Point-to-Point Protocol*) e SLIP (*Serial Line Protocol*), bastando para isso que o programador defina o tipo em que ele está interessado em seu código.

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. O modo promíscuo entretanto é acessível somente ao

usuário supervisor da máquina (conhecido como *root*), conseqüentemente o sistema de captura é executado com prioridade de supervisor.

Então utilizando as funções fornecidas pela *libpcap*, o dispositivo de rede é colocado em modo promíscuo e o sistema de captura começa a receber uma cópia de cada pacote que passa pelo dispositivo, tanto pacotes de entrada como de saída. A partir daí o sistema filtra os pacotes, e armazena apenas as informações mais importantes. As funções da *libpcap* utilizadas neste sistema são:

- *pcap_t* ***pcap_open_live****(char *device, int snaplen, int promisc, int to_ms, char *ebuf)*

pcap_open_live é utilizada para obter um descritor de captura de pacotes para monitorar os pacotes na rede. *device* é uma cadeia de caracteres que especifica o dispositivo de rede a ser aberto. *snaplen* especifica o número máximo de *bytes* a serem capturados. *to_ms* especifica o *timeout* de leitura em milisegundos. *ebuf* é usado para retornar uma mensagem de erro, somente quando a função falha e retorna *NULL*.

- *char* ***pcap_lookupdev****(char *errbuf)*

pcap_lookupdev retorna um ponteiro para um dispositivo de rede apropriado, para ser usado com as funções **pcap_open_live** e **pcap_lookupnet** descrita abaixo. Caso ocorra um erro, a função retorna *NULL* e *errbuf* é preenchida com uma mensagem de erro apropriada.

- *int* ***pcap_lookupnet****(char *device, u_long *maskp, char *errbuf)*

pcap_lookupnet é usada para determinar o endereço e a máscara (*netp* e *maskp*) de rede associados com o dispositivo de rede. Ambos são ponteiros do tipo *u_long*. Um retorno igual a -1 indica um erro e *errbuf* é preenchido com a mensagem de erro apropriada.

- *int* ***pcap_dispatch****(pcap_t *p, int cnt, pcap_handler callback, u_char *user)*

pcap_dispatch é utilizada para processar e coletar pacotes. *cnt* especifica o número máximo de pacotes a serem processados antes de retornar. Se *cnt* for igual a -1 a função processa todos os pacotes recebidos em um *buffer*; se *cnt* for igual a 0 a função processa todos os pacotes até que ocorra um erro ou *EOF*(*End of File*) é atingido. *callback* é um ponteiro para a função que será chamada para tratar os pacotes coletados pela **pcap_dispatch**. Essa função necessariamente deve possuir três parâmetros: um ponteiro para *u_char* o qual é passado pela **pcap_dispatch** (parâmetro *user*), um ponteiro para uma estrutura *pcap_pkthdr* e um ponteiro *u_char* (que é o pacote capturado). O número de pacotes lidos é o retorno da função e um retorno igual a -1 indica um erro.

- *int* ***pcap_compile****(pcap_t *p, struct bpf_program *fp, char *str, int optimize, u_long netmask)*

pcap_compile é usada para compilar a string *str* dentro do filtro do programa. *fp* é um ponteiro do tipo **bpf_program** (estrutura interna da biblioteca libpcap) que será preenchida pela função **pcap_compile.** *optmize* controla se deve ser feita uma otimização no resultado final do filtro do programa *fp*. *netmask* é a máscara da rede local.

- *int* ***pcap_setfilter****(pcap_t *p, struct pcap_pkthdr *h)*

pcap_setfilter é usada para especificar um filtro ao programa que o executa. *fp* é um ponteiro para um vetor de estruturas do tipo **bpf_program**, usualmente o resultado de uma chamada à função **pcap_compile**.

As estruturas *pcap_t* e *pcap_handler* são internas à biblioteca e não serão discutidas aqui. *u_char* e *u_long* são, respectivamente, *unsigned char* e *unsigned long*.

3. Desenvolvimento

O equipamento utilizado para implementação deste protótipo consiste de um IBM-PC *Valuepoint* com processador Intel Pentium 75Mhz, 256kb de memória cache, 16Mb de memória RAM, 1.2Gb de capacidade de disco rígido, e interface de rede *ethernet* de 32 bits em barramento PCI. O sistema foi desenvolvido em linguagem C compilado com compilador GNU C baseado em sistema operacional Linux v2.0.0. Foi escolhido esse ambiente por se tratar de equipamento de fácil obtenção e de baixo custo, além de permitir rápida e fácil portabilidade.

Como já foi mencionado anteriormente, o sistema de captura em discussão é uma implementação parcial do modelo de detecção de intrusão proposto por [CMM+96] e [CMC+96]. Seu funcionamento modular pode ser descrito, em linhas gerais, da seguinte maneira (figura 2):

- Módulo de Captura - captura um fluxo de dados na rede e passa os pacotes ordenados para o módulo de conexão, sob controle do módulo de pré-seleção.

- Módulo de Pré-Seleção e Sistema Especialista - determina e realiza uma filtragem inicial dos eventos que possam representar interesse (tipo de protocolo, serviços, destinos e origens, etc). Os dados filtrados são enviados para análise pelo sistema especialista, baseado em regras de estruturas de decisão [SS92]. Estes dados servem como elementos de análise para monitoração dos eventos de risco, e vão determinar quando o sistema deve começar a capturar todos os dados de uma determinada conexão, para localizar as assinaturas de ataque. Estes eventos são armazenados numa lista de monitoração, que os mantêm durante um tempo definido.

- Módulo de conexão - uma vez que o módulo de pré-seleção e o sistema especialista identificaram os acessos que diferem dos padrões inicialmente aceitáveis, todos os pacotes provenientes da conexão são enviados para o módulo de conexão que os organiza numa relação de causa e efeito, identificando um fluxo de dados unidirecional.

- Analisador semântico e pós-processador - sua função principal é preparar o vetor de estímulo, que conterá os dados que serão analisados pela rede neural. Isto é feito para cada vetor de conexão que está sob monitoramento, analisando os dados e buscando por assinaturas de ataque.

- Rede Neural - analisa o vetor de estímulo de cada conexão, com os respectivos pesos que representam a importância da ocorrência dos eventos, e procura atribuir um grau de suspeita, que representa o estado de segurança da conexão.

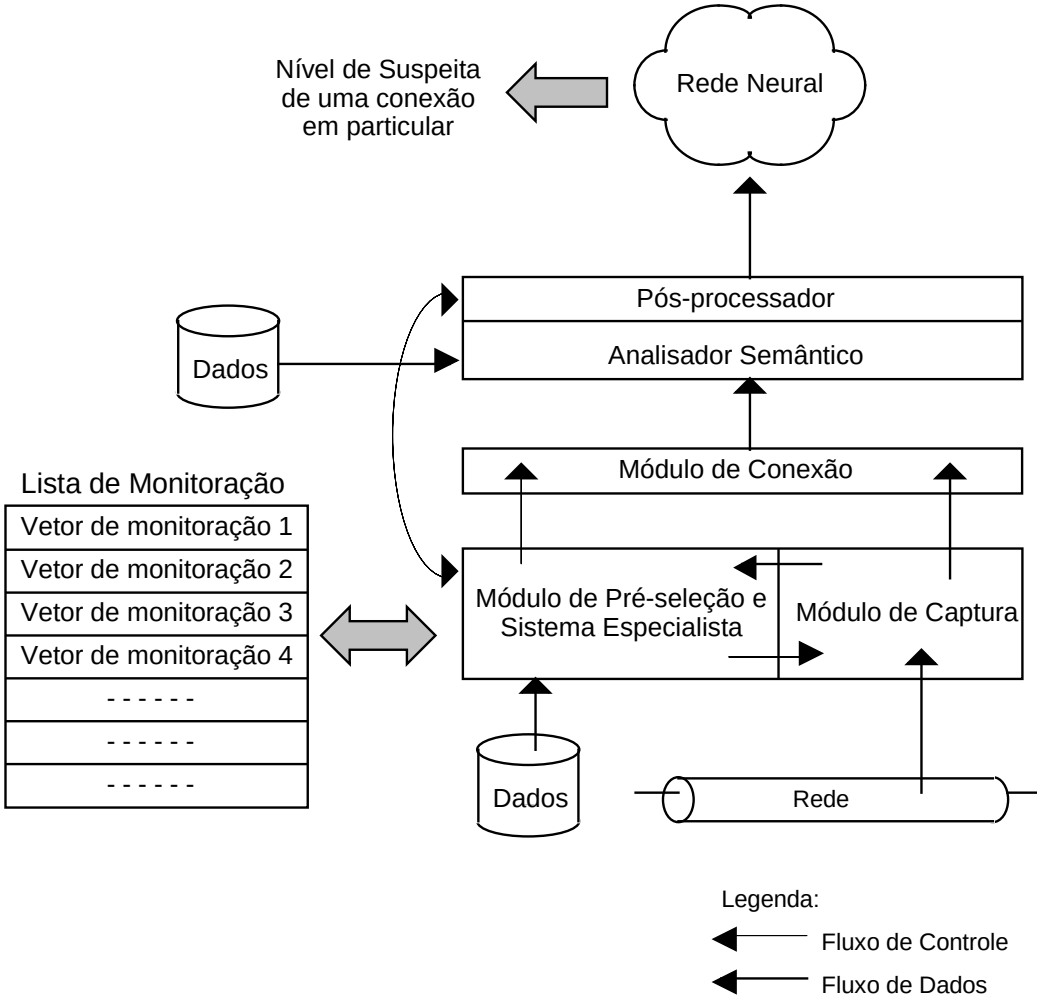

Figura 2 - Estrutura dos módulos do modelo do sistema de detecção de intrusos[CMC+96]

O sistema desenvolvido consiste de um protótipo que engloba o Módulo de Captura, o Módulo de Pré-Seleção e o Módulo de Conexão, porém com um funcionamento menos complexo.

Antes de iniciar o processo de captura de pacotes, o sistema obtém o nome e o endereço IP da máquina onde ele será executado e faz os seguintes testes, nesta ordem:

- verifica se está executando com prioridade de supervisor;
- procura por um dispositivo de rede (função *pcap_lookupdev*);
- inicializa o dispositivo (função *pcap_open_live*);
- obtém o endereço e a máscara da rede (função *pcap_lookupnet*).

Caso algum desses testes falhe, a execução é interrompida imediatamente.

O Modulo de Captura é o primeiro módulo que entra em ação. Utilizando a função *pcap_dispatch* ,ele copia o conteúdo do pacote que está trafegando pela rede para a memória. Em nosso ambiente de desenvolvimento o pacote capturado contém: o cabeçalho ***Ethernet***, o cabeçalho **IP** [Pos81a] e depois o pacote propriamente dito, que pode ser **TCP** [Pos81c], **ICMP** [Pos81b] ou **UDP** [Pos80]. Caso haja algum

problema durante esta operação é chamada uma função específica que fecha o dispositivo de rede (função *pcap_close*) e grava a Tabela de Controle de Conexões, que será discutida mais adiante.

Para a análise de intrusão precisaremos somente dos dados do cabeçalho IP e do pacote que esta encapsulado nele (TCP, UDP e ICMP). Essa parte dos dados deve ser mapeada para estruturas de dados em memória que corresponderão à estrutura do pacote. Isso facilitará o manuseio das informações pelos outros módulos do detector de intrusão.

No caso, o cabeçalho IP é mapeado para a estrutura *IP_header* , apresentada a seguir. O cabeçalho *Ethernet* é descartado durante a operação.

```
struct IP_header {
        unsigned char version_length, type;
        unsigned short length, ID, flag_offset;
        unsigned char TTL, protocol;
        unsigned short checksum;
        unsigned long int source, destination;
        };
```

Os valores mais importantes dessa estrutura são os campos *source* e *destination*, que contém os endereços das máquinas de origem e de destino, respectivamente. O campo *length* contém o tamanho em bytes do cabeçalho IP, utilizado para se determinar exatamente onde termina o cabeçalho e onde começam os dados encapsulados.

Através do valor do campo *protocol* podemos determinar que tipo de dado está encapsulado no pacote e, a partir daí, mapeá-los para a outra estrutura de dados correspondente. Estes dados podem ser do tipo TCP, UDP e ICMP, que serão mapeados respectivamente para as estruturas *TCP_header*, *UDP_header* e *ICMP_header*, descritas a seguir:

```
struct TCP_header {
        unsigned short source, destination;
        unsigned long int seq_nr, ACK_nr;
        unsigned short offset_flag, window, checksum, urgent;
        };
```

Nessa estrutura os campos *source* e *destination* representam as portas da conexão sob monitoração. A partir do valor da porta de destino pode-se determinar que tipo de serviço está em andamento na conexão. Por exemplo, se o destino é a porta 21, trata-se de uma conexão de FTP. A porta de origem pode ser qualquer valor, pois este é gerado em seqüência na máquina de origem. Este valor também é utilizado em nosso tratamento, com finalidade de se diferenciar várias conexões procedentes de uma mesma máquina.

```
struct UDP_header {
        unsigned short source, destination;
        unsigned short length, checksum;
        };
```

Somente os campos *source* e *destination*, representando também as portas de origem e destino respectivamente, são de importância dentro do contexto da captura.

```
struct ICMP_header {
        unsigned char type, code;
        unsigned short checksum;
        };
```

Esta estrutura é utilizada para se monitorar as transmissões de pacotes de ping, a partir dos valores de *type* e *code*.

Dessa maneira tornam-se disponíveis todos os dados que o pacote TCP/IP capturado da rede pode fornecer. Estes dados serão agora passados para o módulo de Pré-Seleção e Sistema Especialista. Em nosso protótipo, para fins de simplicidade, o sistema especialista foi substituído por um pequeno conjunto de regras de decisão.

Antes que este módulo faça uma análise inicial do pacote, este é passado por um filtro BPF instalado no *kernel* pelo programa. Este filtro faz com que só se passe para o nível de programa os pacotes que forem realmente do tipo TCP, UDP e ICMP. Outros pacotes, tipo IPX, SPX, etc., são descartados a nível de *kernel*, segundo a especificação do filtro BPF.

Uma vez que os dados tenham sido submetidos a este filtro, é feita em seguida uma análise do tipo de conexão que este pacote representa. A partir dessa análise poderão ser criadas, ou não, registros na Tabela de Controle de Conexões.

Exemplos de algumas regras adotadas para simplificar a implementação do módulo de pré-seleção:

- Pacotes provenientes da rede interna, ou seja, do mesmo barramento, são descartados;
- Pacotes TCP são tratados se representam uma conexão do tipo *FTP* [PR85], *Telnet* [PR83], *SMTP* (mail)[Pos82], *HTTP* [FGM+97], *Finger* [Zim91], *Pop3*, *exec*, *rlogin*, *rsh* e *lpr*;
- Pacotes UDP que representem conexões do tipo *bootp*, *sunrpc*, *rwho*, *syslog*, *talk*, *route* e *NFS* são analisados;
- Pacotes ICMP que não representem *ping* serão descartados.

A Tabela de Controle de Conexões é uma seqüência de entradas para representar o relacionamento entre uma máquina de uma rede externa na qual o programa de detecção de intrusão está instalado, e a rede interna. Esta tabela está representada na figura 3.

Cada entrada conterá o endereço do *host* de origem (que será de uma rede externa), o qual é o índice da tabela, e uma pequeno vetor com os endereços dos *hosts* internos aos quais o *host* origem fez conexões, com as respectivas portas de origem e destino, tempo inicial (TS) e tempo final (TL) da conexão.

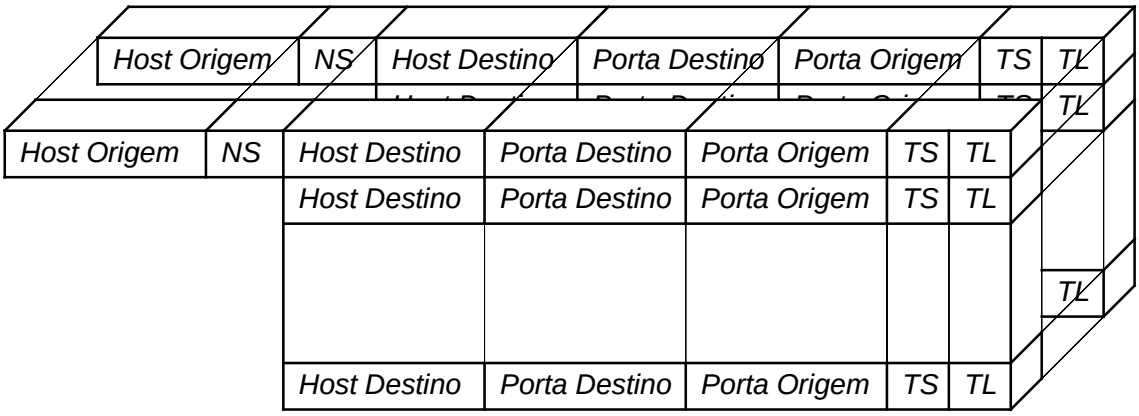

Figura 3 - Tabela de Controle de Conexões

Cada serviço de rede tem o seu nível de segurança próprio. O nível de segurança (representado na tabela por NS) é um valor numérico que vai sendo acrescido à medida que são monitorados eventos de interesse provenientes de uma dada origem (tabela 1). Esses eventos podem ser, por exemplo:

- Uma mesma origem acessando repetidamente um mesmo destino na rede sob monitoramento;
- Um mesmo domínio de origem (ou subconjunto deste domínio) acessando, ou tentando acessar, uma ou mais máquinas destino, na rede sob monitoramento;
- Conexões, ou tentativas de conexão, internas ou externas, ocorrendo em horários não esperados.

Protocolo / Porta	***NS***	***Serviço***
TCP / 21	15	FTP
TCP / 23	15	Telnet
TCP / 25	5	SMTP
UDP / 67	10	Bootp
TCP / 79	2	Finger
TCP / 110	5	Pop3
UDP / 111	10	SunRPC
TCP / 512	5	Exec
UDP / 513	10	Rlogin
TCP / 514	15	Rsh
TCP / 515	10	Lpr
UDP / 520	10	Route
UDP / 2049	15	NFS
Outras	1	Outras portas não listadas (default)

Tabela 1 - Valores de NS, referentes às portas, adotados no módulo de pré-seleção para determinar o disparo automático de captura.

Aos serviços de rede são associados valores padrão para o NS, de acordo com as capacidades de cada um. O *telnet* por exemplo oferece mais capacidade (e portanto maior risco de intrusão) do que o *finger*.

Um exemplo: o sistema detecta um *finger* vindo do *host* A (externo à rede em monitoração) para o *host* B (interno à rede) e inclui um vetor para este evento na tabela de conexões, com um valor de nível de segurança pré-determinado para o *finger* (NS=2). Ao detectar um novo evento, por exemplo um *telnet* (NS=15), a tabela de conexões é pesquisada e, encontrando-se uma entrada proveniente da mesma origem-destino, o nível de segurança deste vetor existente é acrescido do valor do nível de segurança do *telnet*, ficando com NS=17.

Os valores de TS e TL são importantes para se proceder uma análise periódica e cíclica da tabela, para que certas conexões sejam removidas após um determinado tempo de vida, uma vez que se decida que elas não são mais relevantes para a análise de intrusão.

Utilizando os níveis de segurança, o administrador do sistema define um limiar que trata-se do valor máximo que NS pode assumir. Caso o NS de uma conexão ultrapasse esse limiar, esta passa a ser considerada potencialmente perigosa para a rede. Quando isso ocorre é mandado um sinal para o Módulo de Conexão para que comece a registrar todos os pacotes vindos da rede onde esta situada a máquina originadora dessa conexão, sem restrições.

Quando o Módulo de Conexão recebe esse sinal do Módulo de Pré-Seleção, ele cria uma entrada na tabela de domínio suspeitos, onde estarão listados todos os endereços de redes que estão sob suspeita de tentativa de intrusão. Assim que um pacote vindo dos domínios que possuem entrada nessa tabela chegar ao Módulo de Captura, ele será enviado para Módulo de Conexão que se encarregará de gravar em um arquivo os dados dessa conexão. Inicialmente, a partir do caminho pré-definido no módulo, é criado um diretório cujo nome é o endereço IP do domínio onde se origina a conexão (200.100.50.0, p.ex.). Neste diretório, os dados de cada pacote são gravados em arquivos cujos nomes têm o seguinte formato:

PROTOCOLO:IP.origem-TCP.origem_IP.destino-TCP.destino

que dão todas as informações sobre o caminho do pacote. Os pares da conexão podem tanto ser gravados num mesmo arquivo como em arquivos separados. Por exemplo, caso os dados necessitem ser gravados em arquivos separados e estivermos monitorando a conexão do *host* A executando um *telnet* para o *host* B, seriam criados os arquivos *TCP:A-1000_B-23* e *TCP:B-23_A-1000* , que conteriam os tráfegos em cada sentido da conexão. Caso contrário seria criado apenas um arquivo que conteria o tráfego nos dois sentidos, com indicadores especiais que separam os dois fluxos de dados.

4. Ambiente de testes

O sistema de captura de pacotes foi testado em um ambiente de rede bastante heterogêneo, consistindo de máquinas de várias arquiteturas (PC, RISC/6000, Sun-Sparc) e sistemas operacionais (DOS/Novell, AIX, SunOS, Solaris), bem como um tráfego de informações bastante elevado.

Foram utilizados programas para simular comportamentos intrusivos, tais como um produto comercial denominado Internet Security Scan (ISS) e um verificador de falhas de segurança de domínio público, conhecido como SATAN, bem como a simulação manual de alguns ataques específicos [FV93] [Bel89].

Para os testes realizados para a obtenção de assinaturas de ataques foi utilizada a seguinte configuração: três máquinas conectadas em rede com acesso Internet, sendo que duas seriam as máquinas envolvidas no ataque (uma origem e outra destino) e a última executaria uma versão do sistema de captura de pacotes. Essa versão implementava apenas o Módulo de Captura, permitindo a monitoração de todos os dados da conexão onde os ataques eram simulados.

Como exemplo, considere o desenvolvimento de uma conexão TCP com o processo de transmissão de correio eletrônico em protocolo SMTP, descrita a seguir. Esta conexão tenta explorar uma vulnerabilidade antiga, e bastante conhecida, do processo *sendmail* [CA95:02], a fim de capturar o arquivo de senhas do sistema *vitima.algum.lugar.br* e enviá-lo para *intruso@outro.lado.br*:

[conexão tcp estabelecida]
220 vitima.algum.lugar.br ESMTP Sendmail 8.7.5 ready.
mail from " | /bin/mail intruso@outro.lado.br < /etc/passwd "
250 " | /bin/mail intruso@outro.lado.br < /etc/passwd "... sender ok.
rcpt to: nobody
250 Recipient ok.
354 Enter mail, end with "." on a line by itself
data .
250 QAA23003 Message accept for delivery
quit
221 vitima.algum.lugar.br closing connection.
[conexão encerrada]

De posse dessas informações obtidas pelo sistema de captura, o analisador semântico, baseado nos perfis de ataque, pode localizar características mais relevantes da conexão, que correspondem a este tipo de comportamento. Para os testes do sistema completo, com todos os módulos, foi utilizado apenas uma máquina, que monitorava conexões feitas a partir de *hosts* externos à rede.

5. Resultados obtidos e situação atual

A partir dos testes realizados com o protótipo do sistema desenvolvido, obteve-se um conjunto de dados muito positivo que representa o comportamento de diversas técnicas de intrusão, obtidas a partir do monitoramento de conexões com comportamento intrusivo simulado, e de conexões não anômalas. Cerca de 120 tipos diferentes de ataques (e variações) foram capturados e registrados. A análise destes dados teve como resultado a obtenção de um conjunto de assinaturas de ataque, permitindo o desenvolvimento do analisador semântico e o treinamento da rede neural do modelo de detecção de intrusão discutido anteriormente. Atualmente encontra-se em desenvolvimento a interligação dos diversos módulos com um sistema simulador de rede neural, que possa efetivamente detectar, em tempo real, o comportamento intrusivo baseado nas assinaturas de ataque. Além disso estão sendo desenvolvidas ferramentas que permitam interfacear e gerenciar o sistema, além de emitir diversos níveis de alerta ou adoção automática de auditoria ou contramedidas, como por exemplo ajuste ativo de *firewalls* ou filtros.

6. Referências

[Bac94] R. Bace. A New Look at Perpetrators of Computer Crime. In *Proceedings of 16th Department of Energy Computer Security Conference*, 1994.

[Bel89] Bellovin, S.M. "Security Problems in the TCP/IP Protocol Suite". Computer Communications Review, April 1989.

[CA95:02] Computer Emergency Response Team Advisory CA-95:02. "Vulnerabilities in /bin/mail".
URL: ftp://info.cert.org/pub/cert_advisories/CA-95:02.binmail.vulnerabilities

[CMM+96] Cansian, Adriano M.; Moreira, Edson dos S.; Mouro, Rogério B.; Morishita, Fábio T.; Carvalho, André C.P. de L.F. "Um Sistema Adaptativo de Detecção de Intrusão em Redes de Computadores". Notas do ICMSC-USP n°22. Fevereiro 1996.

[CMC+96] A.M.Cansian, E.S.Moreira, A.C.P.L.F. de Carvalho e J.M.Bonifácio Jr. Network Intrusion Detection Using Neural Network. Aceito para International Conference on Computational and Intelligence and Multimedia Applications (ICCIMA'97), 10-12 February, Gold Coast, Australia.

[FGM+97] Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Berners-Lee, T. "Hypertext Transfer Protocol - HTTP 1.1". *Request for Comments* RFC-2068, 1997.

[FV93] Farmer, D.; Venema, W. "Improving the Security of Your Site by Breaking Into it". Sun Microsystems, 1993.

[MCV92] McCanne, Steven; Van Jacobson. "The BSD Packet Filter: A New Architecture for User-level Packet Capture". Lawrence Berkeley Laboratory. December 1992.

[NP89] P. Neumann & D. Parker. A Summary of Computer Misuse Techniques. In *Proceedings of the 12th National Computer Security Conference*, pages 396-407, Washington, DC, USA, October 1989.

[Pos80] Postel, J. B. "User Datagram Protocol (UDP)". *Request for Comments* RFC-768, 1980.

[Pos81a] Postel, J. B. "Internet Protocol (IP)". *Request for Comments* RFC-791, 1981.

[Pos81b] Postel, J. B. "Internet Control Message Protocol (ICMP)". *Request for Comments* RFC-792, 1981.

[Pos81c] Postel, J. B. "Transmission Control Protocol (TCP)". *Request for Comments* RFC-793, 1981.

[Pos82] Postel, J. B. "Simple Mail Transfer Protocol (SMTP)". *Request for Comments* RFC-821, 1982.

[PR83] Postel, J. B.; Reynolds, J. "Telnet Protocol Specifications". *Request for Comments* RFC-854, 1983.

[PR85] Postel, J. B.; Reynolds, J. "File Transfer Protocol". *Request for Comments* RFC-959, 1985.

[SS92] Steven R. Snapp and Stephen E. Smaha. Signature Analysis Model Definition and Formalism. In *Proceedings of the Fourth Workshop on Computer Security Incident Handling*, Denver, CO, USA, August 1992.

[Sun90] Sun Microsystems Inc. "NIT(4P); SunOs 4.1.1 Reference Manual.". October 1990.

[Zim91] Zimmerman, D. "The Finger User Information". *Request for Comments* RFC-1288, 1991.