UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Pós-Graduação em Ciência da Computação

Jorge Luiz Corrêa

Um modelo de detecção de eventos em redes baseado no rastreamento de fluxos

UNESP
2009

Jorge Luiz Corrêa

Um modelo de detecção de eventos em redes baseado no rastreamento de fluxos

Orientador: Prof. Dr. Adriano Mauro Cansian

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UNESP

2009

Jorge Luiz Corrêa

Um modelo de detecção de eventos em redes baseado no rastreamento de fluxos.

Dissertação apresentada para obtenção do título de Mestre em Ciência da Computação, área de Sistemas de Computação junto ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

BANCA EXAMINADORA

Prof. Dr. Adriano Mauro Cansian
UNESP – São José do Rio Preto
Orientador

Profª. Drª. Kalinka Regina Lucas Jaquie Castelo Branco
USP – Universidade de São Paulo

Prof. Dr. Aleardo Manacero Junior
UNESP – São José do Rio Preto

São José do Rio Preto, agosto de 2009.

Agradecimentos

Agradeço primeiramente a Deus por estar presente em minha vida e ter me iluminado durante o desenvolvimento deste trabalho.

Ao meu orientador Prof. Dr. Adriano Mauro Cansian pelos valorosos conhecimentos passados, pela confiança em mim depositada nos últimos 5 anos de orientação acadêmica e amigo nas horas vagas (as poucas que temos).

Aos meus pais Dioraci e Ana Maria por todos os valores e suporte familiar prestados nos últimos 24 anos, em especial neste trabalho pelas ocasiões de patrocínio.

Ao meu irmão Jr pelas diversas palavras de incentivo moral, além de discussões técnicas sobre o assunto.

A minha namorada Tássia por me ajudar durante todo este período e superar os momentos de ausência sempre me incentivando.

Ao Laboratório ACME! de pesquisa em segurança, e todos os seus membros, atuais ou não, em especial aos amigos André Proto e Leandro Arabi Alexandre pelos auxílios durante o desenvolvimento deste trabalho, e aos outros membros que mesmo indiretamente contribuíram para esta pesquisa.

Ao PPGCC – Programa de Pós Graduação em Ciência da Computação – do qual fiz parte do conselho como representante discente por 2 anos.

A todos os docentes do Departamento de Ciências de Computação e Estatística (DCCE) pelas disciplinas e conhecimentos transmitidos durante minha formação.

A todos os amigos que estão sempre juntos.

Sumário

Lista de figuras

Lista de tabelas

Lista de abreviaturas

DoS – Denial of Service
DDoS – Distributed Denial of Service
FIN – representa a finalização de uma conexão TCP (flag FIN, No more data from sender).
FTP – File Transfer Protocol
HTTP – Hypertext Transfer Protocol
ICMP – Internet Control Message Protocol
IETF - Internet Engineering Task Force
IP – Internet Protocol
IPFIX – IP Flow Information Export
IPv4 – Internet Protocol version 4
IPv6 – Internet Protocol version 6
NAT – Network Address Translation
P2P – Peer-to-Peer
RST – representa a reinicialização de uma conexão TCP (flag RST, Reset the connection)
SDI – Sistema de Detecção de Intrusão
SGBD – Sistema Gerenciador de Banco de Dados
SMTP – Simple Mail Transfer Protocol
SPI – Sistema de Prevenção de Intrusão
SSH – Secure Shell protocol
SYN – representa a sincronização dos números de seqüência de uma conexão TCP (flag SYN, Synchronize sequence numbers)
TCP – Transmission control protocol
UDP – User datagram protocol

Glossário

Accounting – contabilidade, verficação de medidas de tráfego para provisionamento de redes.
Backbone – sistema de interligação de redes de elevado desempenho.
Backdoor – porta de comunicação oculta ao usuário, utilizada para que atacantes conectem a um computador remotamente.
Botnets – conjunto de computadores comprometidos controlados por um atacante e utilizados para disparar ataques coordenados.
Bridge – dispositivo de rede capaz de separar dois domínios de colisão, operando na camada de enlace do modelo de referência OSI.
Broadcast - processo de transmissão de uma informação caracterizado pelo envio desta para vários destinatários simultaneamente.
Buffer – região de memória temporária utilizada para armazenar dados e recuperá-los quando necessário.
Buffer overflow – ocasião em que o tamanho de um buffer ultrapassa a quantidade máxima de armazenamento disponível para ele, invadindo posições de memórias de outros processos.
Cache – dispositivo de acesso rápido utilizado como memória temporária.
Crafted packets – pacotes mal formados, cujas especificações não estão de acordo com padrões estabelecidos.
Datacenters – são locais de centralização de dados e redes com estruturas próprias para isto.
Driver – software responsável por intermediar a comunicação entre softwares comuns e o hardware de um computador, ou entre dois softwares.
File sharing – compartilhamento de arquivos.
Flags – São valores que indicam determinada operação/ocorrência no protocolo TCP.
Flash crowd – designação para eventos que geram um aumento acentuado em determinada medida, causado por algum fator externo.
Flooding – inundação, quando vários pacotes de dados são enviados a um destino de uma só vez.
Honeypots – são sistemas de computadores preparados para serem comprometidos, possibilitando que analistas de redes coletem informações sobre as metodologias utilizadas no ataque.
Host – um computador qualquer.
Hub – dispositivo de rede que permite a ligação física de vários computadores em uma rede, gerando um único domínio de colisão.
Gateway – ponto de ligação, é um computador ou dispositivo de rede (como um Firewall ou roteador) responsável por interligar redes diferentes.
Kernel – é o núcleo de um sistema operacional.
Link – enlace de dados, canal de comunicação.
Log – designação do procedimento de geração de registros de ocorrências executados por processos em um sistema.
Malware – programa malicioso, de Malicious Software.
Outliers – pontos discrepantes.
Pattern Matching – reconhecimento de padrões.
Payload – conteúdo de um pacote.
Scan – varredura.
Scanner – o atacante que realiza uma varredura.

Softphone – programa de computador que atua como um telefone, utilizando protocolos de comunicação digital.
Spoofing – mascaramento de endereços, forjamento de dados.
Stub – afunilamento; em redes o equipamento stub é aquele que concentra informações de todo um ambiente.
Tracker – centro de informações de sistemas bittorrent, onde os usuários se conectam para buscar informações dos participantes das redes distribuídas.
Worm – é um programa malicioso capaz de se propagar automaticamente.

Resumo

Este trabalho apresenta um modelo de detecção de eventos em redes baseado no rastreamento de fluxos no padrão Netflow. Atualmente, a utilização de fluxos de rede como mecanismo de monitoria de tráfego tem se tornado cada vez mais importante devido a escalabilidade proporcionada. Inicialmente estabelece-se uma arquitetura de coleta e armazenamento de fluxos baseada em bancos de dados relacionais. Coletados os fluxos, criam-se estruturadas denominadas assinaturas para a descrição dos eventos de interesse no ambiente monitorado. O modelo utiliza duas vertentes na detecção de eventos: a baseada em abuso e a baseada em anomalias. A detecção baseada em abuso visa identificar eventos que produzam características fixas no tráfego de um ambiente. A detecção por anomalias visa identificar padrões de tráfego considerados anormais, podendo utilizar diferentes mecanismos de detecção. A arquitetura do sistema é capaz de coletar e armazenar fluxos, processá-los confrontando-os com uma base de assinaturas, utilizar mecanismos de detecção de anomalias e produzir relatórios para o administrador. O sistema foi testado em um ambiente isolado para coleta de informações, tais como taxas de erros e acertos, e no ambiente de produção do Instituto de Biociências, Letras e Ciências Exatas de São José do Rio Preto (IBILCE - UNESP). Além de eventos isolados de interesse dos administradores, podem ser descritos e detectados eventos como ataques de dicionário, *hosts* com aplicações de compartilhamento de arquivos (P2P), Bittorrent, chamadas de voz Skype, varreduras de redes e artefatos maliciosos. O modelo é aplicável em redes de pequeno e médio porte sem grandes investimentos, permitindo que eventos sejam detectados por meio da identificação de padrões comportamentais que estes geram no ambiente de rede. Testes mostraram que o modelo é capaz de descrever diversos protocolos e padrões de ataques, com taxas de acertos e erros compatíveis com ferramentas de segurança reconhecidamente eficientes.

Abstract

This work presents a detection model of networks events based on the tracking of Netflow standard flows. Currently, the use of network flows as a mechanism for monitoring traffic has become increasingly important because of the scalability provided. Initially an architecture is established for collection and storage of flows based on relational databases. Once collected the flows, structures called signatures are created to describe the events of interest in the environment monitored. The model uses two strands in the detection of events: one based on the abuse and one based on anomalies. The abuse detection aims to identify events that produce fixed characteristics in the traffic of an environment. The anomaly detection aims to identify traffic patterns considered abnormal and may use different mechanisms of detection. The architecture of the system is able to collect and store flows, process them confronting them with a signature database, make use mechanisms of anomaly detection and produce reports for the administrator. The system was tested in an isolated environment for collecting information such as rates of errors and successes, and in the production environment of the Instituto de Biociencias, Letras e Ciencias Exatas de Sao Jose do Rio Preto (IBILCE - UNESP). Further of isolated events of interest of administrators, can be detected and described events as a dictionary attack, hosts with file sharing applications (P2P), BitTorrent, Skype voice calls, network scans and malicious softwares. The model is applicable to networks of small and medium businesses without large investments, allowing events to be detected by identifying behavioral patterns that they generate in the network environment. Tests showed that the model is able to describe several protocols and patterns of attacks, with rates of hits and misses compatible with security tools known efficient.

Capítulo 1 - Introdução

1.1 Descrição do problema

Os responsáveis pela manutenção de um ambiente de rede confrontam-se diariamente com uma diversidade de eventos, tanto de caráter lícito quanto de caráter malicioso. A identificação desses eventos é extremamente importante para manutenção da ordem dentro da infraestrutura de rede, garantindo a prestação dos serviços com qualidade e segurança.

O atual paradigma de tráfego facilita a dissimulação de eventos que interferem diretamente no bom andamento de uma rede. A principal característica para essa dissimulação é a grande variedade de protocolos diariamente utilizados, agregada às altas taxas de dados que produzem. A Internet proporciona acesso a serviços não existentes em décadas anteriores, como voz sobre IP[1] (*Internet Protocol*), transmissões de vídeo e aplicações de compartilhamento de arquivos, serviços esses que possuem uma densidade de tráfego incomparável aos antigos protocolos de chamada de procedimento, navegação e conexões remotas.

Assim como evoluem os protocolos e serviços devem evoluir as metodologias para a manutenção da segurança dos ambientes de rede. A preocupação com a segurança no desenvolvimento das aplicações emergiu recentemente. A maioria dos avanços que surgiram com a disseminação da Internet concentrou-se no aprimoramento de suas funções, ou seja, das novas características que estas aplicações trariam para o mundo real. À segurança, na maioria das vezes, não foi dado o devido zelo, culminando atualmente em regras rígidas para que este quesito esteja presente em sistemas computacionais, na forma de diretrizes e padrões, em um caráter de certo modo forçado e não espontâneo dos desenvolvedores.

Nesse contexto, aplicações independentes e particulares surgiram como imprescindíveis na manutenção da ordem. Nas redes de computadores, veículo pelo qual as aplicações atuais operam, estas ferramentas desempenham papéis como

[1] Para melhor identificação dos termos técnicos utilizados neste texto sob a forma de acrônimos e abreviaturas, consulte a *Lista de Abreviaturas* na página *viii*, ou o *Glossário* na página *ix*.

monitores, ferramentas estatísticas, analisadores de tráfego e filtros. Cada um destes tipos de aplicação cobre um determinado nicho para que a segurança seja estabelecida.

Os sistemas de *Firewall* são simples analisadores de pacotes capazes de bloquear àqueles que estão (ou não estão) de acordo com determinadas regras. Estas regras baseiam-se praticamente na verificação de valores presentes em cada campo de um protocolo, como o IP, TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*). Os Sistemas de Detecção de Intrusão (SDIs) além da possibilidade de verificar campos de protocolos são capazes de analisar o conteúdo de cada pacote que passa pela rede. Existem outros sistemas, como os *Gateways* de aplicação e Sistemas de Prevenção de Intrusão (SPIs), com funcionalidades semelhantes aos anteriormente citados. Todos começam a sentir os efeitos da densidade de tráfego a ser analisada. Firewalls que antigamente interceptavam a rede de uma instituição inteira passaram a ser repensados, principalmente devido às exigências de hardware necessárias. Sistemas detectores de intrusos podem inserir latência em determinados fluxos de dados, como os multimídias, ao analisar cada pacote separadamente.

Começam a surgir então modificações na maneira como lidar com o tráfego de uma rede. Kruegel (Kruegel *et al.*, 2002) propõe um mecanismo de detecção de ataques que se inicia pelo particionamento do tráfego, reduzindo o volume a ser analisado, para em etapa posterior realizar de fato a detecção. Mostra ainda como a performance de um detector centralizado diminui conforme se aumenta a velocidade dos *links* ou o número de assinaturas. De modo geral, o principal objetivo destas mudanças é adaptar os mecanismos de segurança ao padrão atual de tráfego e fornecer certo nível de escalabilidade que acompanhe o avanço gradual das transmissões.

Mesmo que em condições ideais de implantação, superando-se problemas de densidade de tráfego e requisitos de hardware, alguns eventos atuais fogem do escopo dos mecanismos mencionados, tais como alguns tipos de ataque e violação de políticas de segurança e de uso. O surgimento do padrão IPFIX/Netflow (Claise, 2004) (Quittek *et al.*, 2004) de fluxos de rede vem corroborar estas afirmações, estabelecendo uma nova visão de monitoria e proteção de redes a partir de informações de um amplo perímetro, geradas em pontos estratégicos das redes. O que difere os fluxos de rede dos mecanismos mencionados é fonte de informações utilizada para as tarefas de detecção. Os mecanismos anteriormente mencionados utilizam o próprio tráfego como fonte de informação, analisando os pacotes conforme passam por um ponto de monitoria onde estes sistemas estejam posicionados. O conceito de fluxos parte do princípio de

sintetizar informações em pontos estratégicos de uma rede, normalmente considerando um perímetro grande. Implantações padrão para exportação de fluxos ocorrem normalmente em roteadores conhecidos como *stub*, ou seja, o equipamento que liga a rede de uma instituição com outra rede (ou com a Internet). Neste ponto são coletadas diversas informações, notadamente características dos protocolos de camada de rede e transporte do tráfego de entrada e saída. Embora um perímetro maior seja considerado, a quantidade de informações coletadas (a ser analisada) é menor quando comparada à quantidade que o mesmo tráfego produziria se fosse utilizado um SDI convencional. O principal fator que possibilita tal síntese é a não utilização do conteúdo dos pacotes pelos fluxos. Ao não utilizar os *payloads*[2] existe, de fato, uma perda de informações relevantes para detecção de alguns eventos. No entanto, este trabalho mostra que mesmo sem estas informações é possível identificar uma diversidade de eventos analisando *o que eles causam em uma rede*, as características de comunicação entre os *hosts* e o modo de utilização de endereços e portas. Desta maneira, com um mesmo requisito de hardware é possível monitorar e identificar eventos em um perímetro muito maior do que o abrangido por um *Firewall* ou SDI.

1.2 Motivação e objetivos

Nos últimos anos, a busca por novas metodologias de análise de tráfego tem despertado grande interesse entre os pesquisadores da área de segurança. Com a popularização cada vez maior da Internet e com o aumento da velocidade de tráfego disponibilizada ao usuário final, novos eventos passaram a ser alvos constantes de pesquisas. Atualmente, artefatos maliciosos têm a capacidade de se propagarem automaticamente com efeitos até certo ponto catastróficos para uma rede. Nesse período, a análise de tráfego baseada no comportamento do ambiente ganhou grande expressão. Geer (Geer, 2006) mostra em seu trabalho como estas metodologias ganharam popularidade nos últimos anos. No entanto, Geer mostra que a utilização de metodologias de detecção de anomalias com base na análise de comportamento de tráfego não está completamente consolidada e sendo utilizada em produção. Isto se deve a dois fatores principais: tais metodologias geram mais falso-positivos que sistemas

[2] *Payload*: neste trabalho, quando utilizado o termo *payload*, será uma referência ao conteúdo dos pacotes de rede, termo este difundido e utilizado na literatura sobre detecção de intrusão.

baseados em assinaturas convencionais, além de necessitar de maiores requisitos de processamento.

Nesse contexto, pesquisas foram inicialmente desenvolvidas com foco no estabelecimento de nova metodologia para a detecção de anomalias. Diversos trabalhos propunham novos algoritmos e metodologias para detecção de anomalias utilizando fluxos. Como primeiro resultado, foi obtido um modelo de detecção de anomalias baseados em fluxos e redes neurais artificiais (Cansian; Corrêa, 2007). Este trabalho utilizava bibliotecas externas desenvolvidas em linguagem C para acessar os dados contidos nos fluxos. Esta particularidade dificultava o acesso às informações necessárias para alimentar o modelo. Diante da necessidade de acesso versátil às informações dos fluxos e baseando-se em partes no trabalho de Nickless Bill (Bill, 2000) foi estabelecida uma arquitetura de armazenamento de fluxos que, não somente dá suporte ao modelo sendo proposto neste texto, como facilita o desenvolvimento de outras pesquisas que utilizem fluxos de rede (Corrêa; Proto; Cansian, 2008).

De modo geral, as motivações para este trabalho são as restrições de ferramentas anteriormente utilizadas, como os *Firewalls* e SDIs, as necessidades de adaptação dos mecanismos de segurança para o atual paradigma de tráfego e a tentativa de minimizar os problemas das metodologias de detecção por comportamento de tráfego.

O objetivo desta pesquisa é estabelecer um modelo de monitoria de rede capaz de detectar não apenas eventos ilícitos, mas qualquer evento de interesse em redes de computadores, por meio da criação de estruturas descritoras de tráfego, as assinaturas. Toda a metodologia é baseada na utilização de fluxos Netflow. As assinaturas são estruturas capazes de descrever um comportamento dentro do tráfego de um ambiente. No entanto, estas assinaturas podem representar mais de um tipo de evento, não apenas um, como ocorre nas assinaturas dos SDIs. Ainda, dependendo do que novas ameaças causem nas redes, uma assinatura pode ser capaz de detectar novos tipos de ataque que se assemelhem a algum padrão de tráfego já descrito. Como será explanada, a metodologia utilizada nas assinaturas consiste na descrição, passo a passo, de características de tráfego que possam identificar determinado evento. Um objetivo secundário é produzir uma ferramenta que possa ser usada na monitoria de redes de pequeno e médio porte, tornando acessível um dispositivo eficiente de segurança sem a necessidade de grandes investimentos.

1.3 Organização do trabalho

Este trabalho está dividido em cinco capítulos, incluindo este que trata da Introdução. O capítulo 2 trata da revisão bibliográfica contemplando conceitos e definições importantes para o entendimento do trabalho, além dos trabalhos relacionados e do estado da arte sobre pesquisas com fluxos. O capítulo 3 trata do modelo proposto, suas características e modelagens. O capítulo 4 trata dos resultados obtidos nos testes realizados com o modelo. Por fim, o capítulo 5 trata das conclusões e trabalhos futuros.

Após o capítulo final estão dispostas as referências bibliográficas e os trabalhos produzidos como resultados desta pesquisa.

Capítulo 2 - Revisão bibliográfica

Este capítulo introduz todos os conceitos relacionados aos sistemas de monitoria de tráfego em redes de computadores. Tais conceitos são importantes tanto para o posterior entendimento do sistema de rastreamento de fluxos quanto para se ter uma visão geral de quais metodologias de análise de tráfego são utilizadas atualmente.

2.1 A pilha de protocolos TCP/IP

Uma vez que a base deste trabalho é a utilização de fluxos de rede Netflow, é importante conhecer a pilha de protocolos TCP/IP, utilizada na Internet. As informações contidas em cada fluxo relacionam-se com protocolos desta pilha, notadamente em duas camadas dela.

De modo geral a pilha de protocolos TCP/IP é uma organização em camadas de forma que cada uma desempenhe determinada função. Cada camada é responsável por prover serviços a uma camada superior e utilizar serviços de uma camada inferior. Este trabalho considera que o leitor esteja familiarizado com os conceitos básicos da pilha TCP/IP, de forma que eles não serão repetidos aqui. Para esclarecimentos a respeito deste tópico recomenda-se uma leitura de (Stevens, 1993).

2.2 Eventos em redes

Dentro dos objetivos a que se propõe o modelo de rastreamento de fluxos é importante conhecer os principais tipos de evento que ocorrem em uma rede, sejam eles lícitos ou não. A maior parte desta seção trata dos eventos ilícitos, normalmente ataques. A subseção 2.2.6 discute a importância de se monitorar eventos lícitos. No entanto, existem também vários outros eventos de segurança mais relacionados com SDIs baseados em *hosts*, ou seja, não fazem parte do escopo dos objetivos do sistema de rastreamento de fluxos. Dentre estes eventos pode-se citar a exploração de vulnerabilidades, ataques de *buffer overflow* em memória local e códigos maliciosos que executam tarefas locais.

2.2.1 Mapeamento de rede

O mapeamento de rede normalmente precede outros tipos de ataque mais específicos, sendo considerado como a fase de reconhecimento do ambiente alvo. Trata-se do processo de descobrir *hosts* ativos em uma rede. São também conhecidos como *scans*, varreduras ou tentativas de prospecção (extração de informações). Detectar um mapeamento de rede pode auxiliar administradores na defesa de um ambiente. Pressupondo que após este evento um ataque pode ocorrer, um administrador pode elevar sua atenção para tráfegos cuja origem é a mesma do '*scanner'*.

O resultado de uma varredura permite a um atacante definir os alvos ativos e prosseguir com a tarefa de elicitação de informações. Um próximo passo é descobrir quais tipos de serviço um *host* que está ativo executa.

2.2.2 Mapeamento de serviços

O mapeamento ou *scan* de serviços é o levantamento de quais serviços são executados em determinado *host*. Tanto pode suceder o mapeamento de rede quanto pode ser executado como primeiro passo. O exemplo clássico da utilização deste evento é na determinação de quais *hosts* executam determinado serviço, quando se conhece uma maneira de explorar uma vulnerabilidade deste serviço.

Embora na maioria das vezes os mapeamentos estejam relacionados com eventos maliciosos, podem ser utilizados como ferramentas de auditoria de sistemas, permitindo que os responsáveis por determinados sistemas adquiram informações sobre a corretude dos serviços que devem manter.

2.2.3 Negativa de serviço para esgotamento de recursos

Os ataques de negativa de serviço (conhecidos como DoS, de *Denial of Service*) têm por objetivo fazer com que recursos sejam negados para usuários legítimos. No caso de DoS por esgotamento de recursos, o alvo do ataque pode ser tanto um *host* específico quanto um serviço específico em um *host*.

Existem diversas maneiras de se efetuar um DoS. Muitas delas utilizam-se de falhas de protocolos ou de software e a partir de pacotes mal formados (conhecidos como *crafted packets*) fazem com que o serviço passe a ser negado a usuários legítimos. Por exemplo, eram comuns nos princípios da Internet, serviços suscetíveis a ataques de

spoofing (falsificação) de endereço origem. Um atacante construía um pacote mal formado colocando o mesmo endereço como origem e destino. Ao receber o pacote o serviço gerava uma resposta para o próprio *host* que o executava. Ao recusar a resposta o *host* entrava em um laço, gerando respostas de recusa para si mesmo, fazendo com que recursos como a memória fossem consumidos, passando a negar serviços para requisições legítimas.

O mesmo tipo de problema ocorria com serviços cuja programação não previa o que executar quando determinados dados ocorressem em determinados campos dos protocolos. Assim, serviços abortavam quando, por exemplo, determinados dados eram colocados em determinadas *flags* do TCP ou no campo *Opções*.

Do ponto de vista da monitoria de rede, o ataque de *SYN Flood* é responsável pela geração de DoS em alguns serviços. A metodologia consiste em abrir diversas conexões (*flag SYN* do TCP) para determinado serviço fazendo com que o *host* reserve recursos como memória e *buffers*. Em determinado momento este *host* passa a recusar solicitações legítimas para o serviço.

2.2.4 Negativa de serviço para esgotamento de banda

O ataque de negativa de serviço para esgotamento de banda possui o mesmo objetivo discutido na seção anterior. No entanto, em vez de vitimar apenas um serviço ou *host*, tem a capacidade de fazer com que uma rede inteira passe a negar serviços. Isto ocorre esgotando a capacidade de comunicação do *link* da rede alvo.

Este DoS normalmente possui o chamado fator de amplificação. Esgotar os recursos de comunicação de uma rede requer o emprego de mais esforços do que os necessários para esgotar os recursos de um único *host*. Para tanto, atacantes utilizam-se de mecanismos de amplificação de tráfego. Dois deles são a vulnerabilidade a *spoofing* e *botnets*.

Quando uma rede é vulnerável a ataques *spoofing* ela não realiza checagens sobre o endereço de origem e destino dos pacotes. Um atacante pode colocar o endereço de *broadcast* de uma rede como sendo o destino do pacote e o endereço do alvo como sendo o endereço de origem. Desta forma, todos os *hosts* da rede gerarão tráfego (mesmo que inválido) para o *host* alvo, ocupando assim o canal de comunicação. O sucesso do ataque se dá quando a amplificação atinge o limite da capacidade de transferência do alvo. Novas requisições para a rede serão negadas.

As *botnets* são conjuntos de computadores comprometidos e controlados por um atacante, de modo que a qualquer momento este pode lançar ataques a partir de cada um dos computadores. Desta forma, uma *botnet* é um grande artifício de amplificação de ataques, já que diversos *hosts* colaborando podem gerar taxas de tráfego bastante altas.

Conforme explicado, alguns DoSs envolvem vários *hosts* tentando comprometer um ou mais *hosts*. Esta organização é conhecida como *ataque de negativa de serviço distribuído* (*DDoS, Distributed Denial of Service*).

Infelizmente, não há atualmente uma maneira efetiva de se proteger contra os efeitos de um DDoS. Assim, torna-se importante identificar a ocorrência deste tipo de ataque com informações relevantes das origens a fim de se aplicar contra medidas, mesmo que paliativas, mitigantes ou até mesmo políticas.

2.2.5 Ataques exponenciais – *worms*

Os ataques exponenciais são realizados por códigos maliciosos que possuem autonomia para se propagarem, ou seja, a disseminação destes artefatos ocorre de maneira automática independente da vontade do usuário. Se o artefato realiza tudo automaticamente é denominado *worm*. Caso o artefato seja transmitido anexado a outros vetores de propagação é chamado *vírus* (Cheswick; Bellovin; Rubin, 2005).

Os efeitos dos ataques exponenciais são parecidos com os DoS e DDoS. Dependendo do grau de disseminação redes inteiras podem passar a negar serviço durante a propagação destes programas, ocupando recursos de *host* (memória), de rede (banda) e até mesmo pessoal (tempo despendido com a remoção). Em determinados casos os *worms* se tornam mais preocupantes, principalmente quando a disseminação impacta financeiramente no ambiente, como é o caso de serviços de hospedagem ou comércio eletrônico. Assim como as negativas de serviço, é muito importante detectar a disseminação no início, para que as contra medidas sejam tomadas tão logo quanto possível. Do ponto de vista da monitoria, ataques exponenciais normalmente são detectados com sistemas de anomalia, mas determinados artefatos podem ser descritos, configurando assim uma detecção por abuso.

2.2.6 Eventos lícitos

Normalmente sistemas de segurança despendem grandes esforços para detectar eventos de natureza proibida. No entanto, é importante que estes sistemas possam

identificar qualquer tipo de evento, não apenas aqueles que caracterizem uso indevido de um ambiente computacional. Os eventos lícitos são todos os eventos permitidos dentro de uma rede. Por exemplo, se uma instituição permite que seus usuários naveguem pela Internet, acessos HTTP (*Hypertext Transfer Protocol*) a servidores externos são eventos lícitos.

A licitude de um evento será normalmente definida por políticas de uso e de segurança. Desta forma, é possível entender que um mesmo evento será considerado lícito em determinado ambiente, mas ilícito em outro. Por exemplo, se uma instituição permite o uso de aplicações de compartilhamento de arquivos (*file sharing*) este passa a ser lícito, enquanto na maioria das outras instituições este evento será considerado proibido.

Uma das características do sistema de rastreamento de fluxos é justamente permitir que administradores monitorem eventos lícitos, da mesma maneira que monitoram os ilícitos. Estas informações podem ser úteis em diversos sentidos tal como no suporte a tomada de decisões para projetos de rede. Este tipo de análise de tráfego é conhecido como contabilidade (*accounting*).

O grande desafio está na identificação de eventos lícitos que se assemelham a algum tipo de evento ilícito. A classe de eventos conhecida como *Flash Crowd* é um exemplo clássico. O *Flash Crowd* é um aumento acentuado no número de acessos a um serviço ou rede. Isto ocorre normalmente em *sites* que publicam algum conteúdo de grande popularidade. Do ponto de vista da análise de rede, os *Flash Crowds* são muito semelhantes aos DDoS. O baixo índice de falso-positivos para *Flash Crowds* é uma importante métrica para analisar a eficiência de um sistema de monitoria de rede.

2.3 Sistemas de segurança em redes

O número de usuários mal intencionados é cada vez maior na Internet. Para prover certo nível de segurança existem atualmente diversos sistemas utilizados para a análise e monitoria de uma rede, operando em diversas camadas da pilha TCP/IP, com diferentes metodologias de detecção e objetivos. A diferenciação inicia pelo objetivo de cada um deles. Quando uma rede se torna pública, os administradores devem passar a se preocupar com dois pontos cruciais nos projetos de sistemas de segurança.

O primeiro ponto, considerado atualmente o mais valioso, é a proteção dos dados do ambiente. Esta proteção se estabelece de diversas maneiras, como no controle

de acesso a estes dados, na privacidade que determinadas informações requerem ao trafegar por uma rede, na disponibilidade e na integridade dos dados, ou seja, na garantia de que são dados válidos, não alterados ou violados.

O segundo ponto refere-se aos recursos do ambiente. Uma rede ligada à Internet possui diversos recursos tais como computadores, equipamentos de rede e outros tipos de dispositivos que devem operar fornecendo o máximo de sua capacidade. A degradação desta capacidade por fatores externos, tais como ataques ou propagação de artefatos maliciosos, é considerada um evento de segurança.

Este dois pontos referem-se a "*o que proteger*" em um ambiente de rede. Os fatores externos citados anteriormente são diferentes entre si. As metodologias para detectar uma intrusão e um ataque de negativa de serviço são normalmente diferentes. Portanto, existem diversos sistemas de segurança em redes, responsáveis por diferentes escopos de proteção e especializados na detecção de determinados eventos.

Nas seções a seguir são apresentadas as principais categorias de sistemas de segurança em redes. Todas elas devem operar colaborativamente para a obtenção de um ambiente seguro.

2.3.1 Firewalls - filtragem de pacotes

Um *Firewall* é uma das técnicas de segurança mais utilizada em redes de computadores. Atualmente, tornaram-se tão comuns que a maioria dos sistemas operacionais já possui um *Firewall* embutido. No entanto, do ponto de vista de uma rede de computadores, os *Firewalls* normalmente são sistemas posicionados no ponto de conexão com a Internet. A Figura 1 mostra um exemplo de posicionamento comum de um *Firewall*.

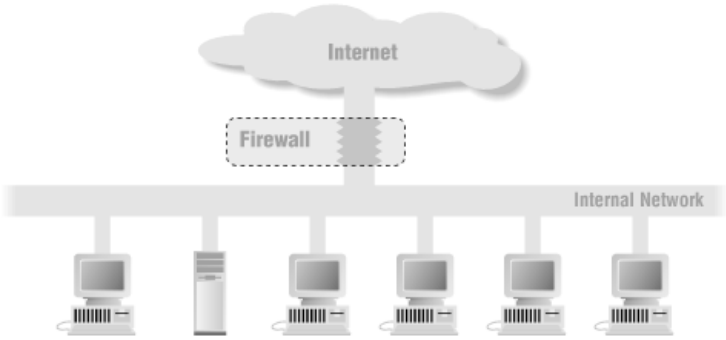

Figura 1. Posicionamento comum de um Firewall separando uma rede interna da Internet.
Fonte: (Chapman; Zwicky, 1995), p.18.

No entanto, nem sempre é possível posicionar um *Firewall* como mostrado, principalmente devido ao tamanho da rede sendo protegida. Como discutido na Introdução, pequenas e médias redes possuem uma densidade de tráfego bastante grande. Utilizar um único *Firewall* pode ser inviável quando pensando em soluções comuns de baixo custo. A solução mais utilizada atualmente é a distribuição destes sistemas, de forma que cada parte de uma rede possua seu próprio sistema de *Firewall*, o que facilita a utilização de políticas de segurança diferentes em locais diferentes de uma rede.

Dentre as funcionalidades que um *Firewall* pode prover estão (Chapman; Zwicky, 1995):

- **Estabelecimento de um ponto de tomada de decisão:** dependendo do posicionamento, um Firewall pode constituir um ponto importante para se decidir que tipos de acessos são permitidos ou não entre as redes as quais ele atua.
- **Fortalecimento de políticas de segurança:** atua como guardião de quais serviços podem ser acessados “na” Internet e quais serviços podem ser acessados “da” Internet.
- **Levantamento de atividades ocorridas na rede:** os *Firewalls* são importantes para a coleta de informações sobre os acessos ocorridos em uma rede, tanto externos como internos (atividades de *log*).

Analisando como um *Firewall* atua, este sistema realiza a filtragem de pacotes segundo informações contidas nos cabeçalhos dos protocolos, comumente os da pilha TCP/IP. Destacam-se o protocolo IP, TCP e UDP como principais no estabelecimento de políticas de filtragem. Em alguns casos podem ser utilizados protocolos da camada de enlace e da camada de aplicação, mas apenas em ocasiões específicas de filtragem (nestes casos os *Firewalls* operam semelhantemente a outros mecanismos de segurança, como os *IDSs* e *Gateways* de aplicação).

Cada pacote analisado por um *Firewall* é comparado com uma lista de regras. Cada regra é composta por detalhes de protocolos e uma ação a ser tomada. Quando existe uma correspondência (*match*) de um pacote com uma regra, a ação é executada. A Figura 2 mostra uma regra de dois conhecidos sistemas de *Firewall*, o Iptables

(Linux) e o IPFW (FreeBSD), responsável por liberar o acesso ao serviço HTTP do *host* 192.168.1.2, acessado da rede 10.0.0.0/8.

```
Iptables:
iptables -A INPUT -s 10.0.0.0/8 -d 192.168.1.2 -p tcp --dport 80 -j ACCEPT

IPFW:
ipfw add 100 permit tcp from 10.0.0.0/8 to 192.168.1.2 80
```

Figura 2. Regras para permissão de tráfego HTTP no Iptables e no IPFW.

As regras de um *Firewall* são do tipo *ACEITA* ou *RECUSA*. Desta forma é possível estabelecer se um pacote é descartado (*RECUSA*) ou não (*ACEITA*). Se um pacote não corresponde a nenhuma regra, a *política* do *Firewall* é analisada. Esta política é semelhante aos tipos de regra, ou seja, o pacote será aceito ou não. Do ponto de vista da construção das regras, normalmente é aconselhado ao administrador bloquear todo tipo de tráfego (política de recusa) e liberar apenas aqueles permitidos pela política.

Portanto, o papel principal de um *Firewall* do ponto de vista da segurança é reconhecer correspondências entre dados dos cabeçalhos dos pacotes e as regras estabelecidas, aceitando ou não o pacote. Esta simples operação de reconhecimento de correspondências restringe o escopo de proteção. Assim, pode-se também elencar tarefas não desempenhadas pelos *Firewalls* (Chapman; Zwicky, 1995):

- **Não proteção contra usuários maliciosos internos:** um Firewall é simplesmente capaz de filtrar tráfegos característicos de alguns protocolos. Assim, é possível proibir que determinados dados de uma instituição saiam da rede interna. No entanto, um usuário interno pode utilizar outros meios para retirar estas informações da instituição.
- **Não proteção contra tráfegos que não passam pelo Firewall:** obviamente, tráfegos que não passam pelo Firewall não se submetem às suas regras. É cada vez mais comum a ocorrência de problemas de segurança devido a conexões estabelecidas de dentro de uma rede, tais como conexões discadas (*dial-up*), sem fio via operadoras de telefonia (modems 3G), ou mesmo pela má instrução de um usuário que instala um ponto de acesso sem fio sem as devidas restrições, possibilitando acesso de usuários não autorizados à rede interna.

- **Não proteção contra vírus e *malwares*:** novamente, os Firewalls são responsáveis por filtrar pacotes utilizando dados como endereços e portas, não detalhes sobre o conteúdo dos pacotes, o que pode caracterizar vírus e *malwares*. Uma possibilidade de analisar conteúdos são os *Firewalls* anteriormente mencionados (camada de aplicação), mas que não são utilizados na prática para esta função, principalmente devido à carga computacional exigida. Além disso, os *malwares* atuais são capazes de utilizar diversas técnicas de ofuscação de dados diminuindo a eficiência deste tipo de análise.
- **Não proteção contra determinados ataques:** diversos eventos não são detectáveis por *Firewalls*, como os ataques de negativa de serviço. Estes eventos necessitam outros mecanismos de detecção, contemplados por outros sistemas de segurança.

2.3.2 Sistemas de detecção de intrusão (SDIs)

A segurança de sistemas computacionais é fundamentada sobre cinco conceitos principais: integridade, autenticidade, confidencialidade, disponibilidade e não repúdio.

- **Integridade:** garantia de que informações estão verdadeiramente válidas, não sofreram qualquer tipo de alteração.
- **Autenticidade:** garantia sobre a veracidade da identidade dos entes comunicantes.
- **Confidencialidade:** garantia de proteção das informações de modo que apenas os entes permitidos consigam acessá-la.
- **Disponibilidade:** garantia de que um sistema estará prontamente disponível para executar suas funções.
- **Não repúdio:** garantia de responsabilidade final, de forma que não seja permitido a um ente comunicante repudiar sua participação em determinados eventos.

Uma intrusão é qualquer tentativa de violar algum destes conceitos. Por exemplo: um usuário que tenta acessar informações as quais não tem permissão está cometendo uma tentativa de intrusão; outro exemplo é a tentativa de tornar um sistema indisponível. Diversos são os objetivos de se realizar uma tentativa de intrusão, como por exemplo, obter informações confidenciais de outros usuários ou empresas, senhas,

instalar programas maliciosos, interesses comerciais ou governamentais ou mesmo sem objetivos planejados, apenas por curiosidade ou disputas entre grupos maliciosos.

Os SDIs são complementares aos *Firewalls* na busca de um ambiente seguro. São sistemas capazes de detectar tentativas de intrusões analisando não apenas cabeçalhos de pacotes (pacotes mal formados), mas também o conteúdo que estes transportam. Existem vários tipos de SDIs classificados de acordo com a metodologia de análise utilizada, com o tempo em que a análise ocorre, e de acordo com a fonte de informações dos eventos (Escamilla, 1998).

Quanto à maneira utilizada para tomar decisões um SDI pode ser:

Estatístico (ou detector de anomalias): possui como metodologia a procura por desvios de medidas estatísticas, a fim de detectar comportamentos não usuais. De modo geral, esta metodologia utiliza variáveis que definem pontos de interesses a serem monitorados. Utilizando dados pré-armazenados são estabelecidos os valores normais para tais variáveis, além de limiares para a detecção. Quando a variável sendo monitorada apresenta valores maiores que o usual a detecção é confirmada. Por exemplo, uma variável pode monitorar o número de autenticações que um determinado usuário realiza em um sistema. Se o número de tentativas exceder o normal, uma tentativa de intrusão é detectada. Dentre as vantagens estão a escalabilidade, baixo uso de memória e facilidade de entendimento de limiares por parte dos administradores. Dentre as desvantagens estão a não detecção de ataques silenciosos (os que não causam distúrbios) e a dificuldade na definição dos limiares que definem os eventos.

Detector de padrões conhecidos (ou baseado em abuso): conhecido na literatura como sistemas *Pattern Matching*, utiliza um conjunto de padrões conhecidos de intrusão que é confrontado com as atividades do sistema (eventos que estão ocorrendo). Se algum destes eventos conhecidos ocorrer novamente a intrusão é detectada. A idéia principal é manter-se protegido contra ameaças já conhecidas. As codificações dos padrões normalmente são chamadas 'assinaturas de ataque'. Dentre as vantagens estão a possibilidade de configuração aprimorada para monitorar apenas o necessário e não desperdiçar recursos (por exemplo, pode-se verificar as assinaturas de ataque contra servidores WEB apenas quando o tráfego tem como destino um servidor WEB, não sendo necessária a verificação quando o destino é um servidor de correio eletrônico) e eficiência no reconhecimento, uma vez que normalmente faz checagem de conteúdo. Dentre as desvantagens estão a dificuldade de escalabilidade, uma vez que o uso de memória é restritivo para redes e tráfegos de grande porte, além da dificuldade

de reconhecer novos eventos e da quantidade de alertas emitidos, requerendo experiência por parte do administrador de rede.

Cada uma destas classes apresenta vantagens e desvantagens, o que as tornam complementares. Em um projeto de segurança que envolva SDIs, é bastante provável que se implemente estas duas classes de SDIs a fim de abranger as diversas possibilidades de eventos existentes.

De acordo com o tempo em que a análise ocorre, os SDIs podem ser:

De tempo real: o sistema realiza reconhecimentos sobre o tráfego corrente de um ambiente, em tempo hábil de se tomar uma decisão que bloqueie ações maliciosas. Em outras palavras, os SDIs permitem que um ataque seja detectado e bloqueado antes de obter sucesso. Por exemplo, em uma tentativa de acesso não permitido a determinados arquivos, o SDI de tempo real bloqueará o atacante antes que este acesse de fato os dados. O grande problema desta categoria é o alto consumo de recursos, tanto de hardware quanto de software. No entanto, são fortemente indicados quando se tem uma quantidade restrita de eventos para monitorar.

Baseado em intervalos: nesta categoria as análises ocorrem em períodos de tempo, e não a todo momento, sobre o tráfego corrente. Embora ocorra uma diminuição na necessidade de recursos computacionais, esta categoria passa a utilizar o paradigma *post-mortem* de análise, ou seja, a detecção pode ocorrer depois que um ataque tenha obtido sucesso. Embora possa parecer ineficiente, a análise *post-mortem* pode ser eficiente para vários tipos de evento, como é o caso do rastreamento de fluxos. Neste caso específico a análise ocorre em intervalos devido a necessidade de se agrupar certas quantidades de fluxos para que seja possível a descrição de um evento. Como muitos desses eventos são duradouros a análise baseada em intervalos mostra-se eficiente. A grande desvantagem da análise baseada em intervalos é a não possibilidade de bloqueio *‘on the fly’* (em andamento, em tempo real) de um evento de curta duração. Para eventos duradouros esta característica extingue-se. Há de ficar claro que o termo duradouro refere-se ao tempo necessário do intervalo entre duas análises consecutivas. Assim, se um sistema executa a cada 1 minuto, um evento pode ser considerado duradouro se perdurar por mais de 1 minuto. Quanto menor este intervalo, mais o sistema se aproximará da análise em tempo real e maiores serão as chances de se conseguir bloquear o evento, mesmo utilizando análise baseada em intervalos.

Por fim, quanto à fonte de informações, um SDI pode ser:

Baseado em *host*: as informações utilizadas são obtidas de um sistema operacional, representando as atividades executadas exclusivamente neste sistema. Normalmente estas atividades são obtidas do núcleo do sistema (*kernel*) por meio de serviços de *log*. O SDI realiza as detecções com base nas informações geradas pelas aplicações e núcleo do sistema, armazenadas nos arquivos de *log*. Portanto, os SDIs baseados em *hosts* atuam no perímetro de um único sistema. Dentre as vantagens estão a possibilidade de analisar ataques provenientes de usuários do próprio *host* e, por atuar no nível do *kernel*, poder analisar ataques mesmo que em canais criptografados, pois podem acessar os dados sem criptografia. Dentre as desvantagens estão a necessidade de se estabelecer uma política consistente de geração de *logs* e a dificuldade em analisar resultados, visto que a quantidade de informações geradas pode ser extremamente grande.

Baseado em rede: utiliza como fonte de informações o próprio tráfego de rede. Normalmente estes sistemas operam com interfaces de rede no modo promíscuo, de forma que todos os pacotes que passam pelo segmento de rede ao qual a interface está conectada são capturados. Quanto às vantagens, os SDIs baseados em redes podem detectar grande parte dos eventos não detectados por SDIs baseados em *host*. Uma das maiores desvantagens dos SDIs baseados em rede é a não possibilidade de analisar tráfego criptografado. Como atualmente diversas aplicações utilizam-se de criptografia, tornou-se um desafio analisar esse tipo de tráfego para detectar intrusões. Como será mostrado a seguir, o sistema de rastreamento de fluxos sendo proposto pode ser classificado como baseado em rede. No entanto, traz uma nova característica que é a detecção de evento sem a análise de conteúdo, ou seja, independente de o tráfego ser criptografado ou não.

A Figura 3 resume a classificação dos SDIs, destacando onde se encaixa o sistema sendo proposto neste trabalho.

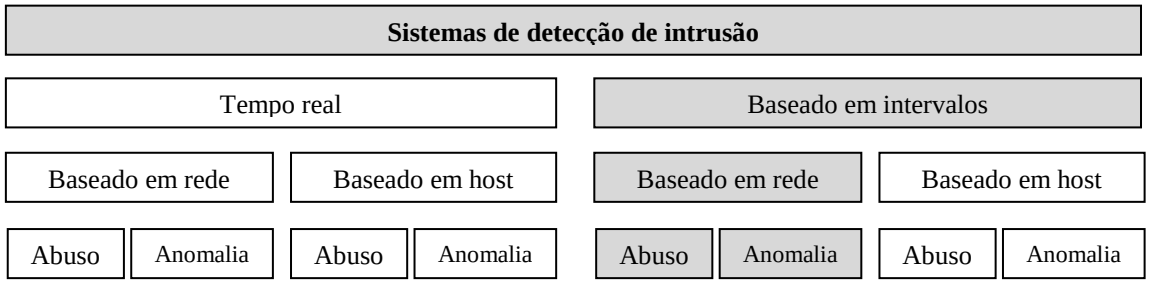

Figura 3. Classificação de um sistema de detecção de intrusão.

2.3.3 Sistemas de prevenção de intrusão (SPIs)

Recentemente, além dos conhecidos SDIs, surgiram os chamados Sistemas de Prevenção de Intrusão (SPIs) (Larsen; Galt; Richardson, 2009) (Araújo; Filho, 2009). Um SPI é um sistema que combina as características de *Firewall* e de um SDI. Esta combinação possibilita diferenciar um SPI de SDI de forma que o primeiro é capaz não apenas de identificar o evento, mas tomar decisões e evitar que pacotes referentes ao evento continuem transitando pela rede. Um SPI é considerado proativo. O método utilizado para isto é estender a análise de um Firewall até a camada de aplicação e utilizar dos mecanismos dos SDI para analisar o conteúdo dos pacotes (Xinyou; Chengzhong; Wenbin, 2004).

Esta metodologia tem gerado diversas ferramentas tanto comerciais quanto de domínio público. O Snort (Roesch, 2009), por exemplo, é considerado o SDI aberto mais difundido no mundo. Ele utiliza uma base de assinaturas atualizável pela Internet capaz de descrever uma grande quantidade de ataques e outros tipos de evento, como possíveis infrações de políticas de uso. Uma modificação do Snort é o Snort Inline (Metcalfe; Julien, 2009). Trata-se de uma versão modificada do Snort capaz de receber pacotes dos sistemas de Firewall Iptables (do Linux) e IPFW (do FreeBSD). De posse dos pacotes, é capaz de utilizar as regras do próprio Snort para detectar eventos podendo então decidir se deve aceitar ou rejeitar o pacote.

2.3.4 Gateways de aplicação

Os *gateways* de aplicação atuam na camada de aplicação do modelo TCP/IP. Estes sistemas possuem conhecimento detalhado de cada protocolo a ser analisado na camada de aplicação (Cheswick; Bellovin; Rubin, 2005). Por exemplo, um *gateway* de aplicação conhece os detalhes do protocolo HTTP (*Hypertext Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) e FTP (*File Transfer Protocol*). Desta forma, é capaz de identificar e entender tráfegos contendo dados desses serviços e aplicar seus filtros com base neste conteúdo. Diferentemente de assinaturas, os filtros dos *gateways* de aplicação normalmente refletem políticas de segurança. Assim, é possível permitir tráfego HTTP normalmente, mas pode-se restringir os pacotes que contenham determinadas palavras. Segundo o mesmo princípio, um usuário pode baixar um arquivo via FTP que será verificado no *gateway* de aplicação quanto à ocorrência de vírus ou códigos maliciosos.

O grande desafio dos *gateways* de aplicação é filtrar protocolos proprietários. Soluções comerciais possuem grandes quantidades de protocolos descritos sob acordos de não divulgação de informações com os reais proprietários do padrão. A grande vantagem destes sistemas é conseguir analisar em um ponto único diversos protocolos, controlando assim tudo o que entra ou sai de uma rede por meio destes.

2.4 Netflow e IPFIX

Diversos interesses implicam na necessidade de medições de tráfego dentro de uma rede de computadores. Estas medições podem ser realizadas por diferentes equipamentos posicionados estrategicamente dentro da topologia do ambiente analisado. Além disso, diversas técnicas podem ser utilizadas, como análises de *logs* nos *gateways* ou captura de pacotes em determinados pontos da topologia. Ciente desta necessidade, o IETF (*Internet Engineering Task Force – Internet Society*), órgão regulador de padrões para Internet, propôs a criação de um padrão cuja finalidade era estabelecer uma arquitetura para análise de tráfego. Este trabalho foi publicado no RFC 3917 (Quittek *et al.*, 2004) contendo toda a especificação de quais características o padrão deveria conter. Subseqüentemente a esta publicação, os padrões já existentes começaram a se adequar à proposta do IETF e outros novos começaram a surgir. O Netflow - RFC 3954 (Claise, 2004), da *Cisco Systems*, é um exemplo. Versões iniciais começaram a ser modificadas e culminaram na versão 9, a mais atual, considerada como adequada às exigências do IPFIX.

Um fluxo Netflow é definido como uma seqüência unidirecional de pacotes entre dois *hosts*, ou seja, é a representação de um tráfego com características comuns. Pode-se entender um fluxo como uma tupla na qual as seguintes informações aparecem com o mesmo valor: endereço IP de origem e de destino; porta de origem e de destino (referente ao protocolo da camada de transporte); valor do campo *Protocol* do datagrama IP; *byte Type of Service* do datagrama IP; e interface lógica de entrada do datagrama no roteador ou *switch*. Estes campos permitem que um fluxo represente concisamente o tráfego através de um ponto de observação (normalmente um roteador). Todos os pacotes com um mesmo valor nos campos desta tupla têm informações contabilizadas em um registro de fluxo, ou simplesmente fluxo, mantidos na memória do equipamento exportador. Os fluxos mantidos no Netflow *cache* são exportados nas seguintes situações: permanece ocioso por mais de 15 segundos; sua duração excede 30

minutos; uma conexão TCP é encerrada com a *flag* FIN ou RST; a tabela de fluxos está cheia (no equipamento) ou o usuário redefine as configurações de fluxo. Um exemplo deste sistema é mostrado na Figura 4.

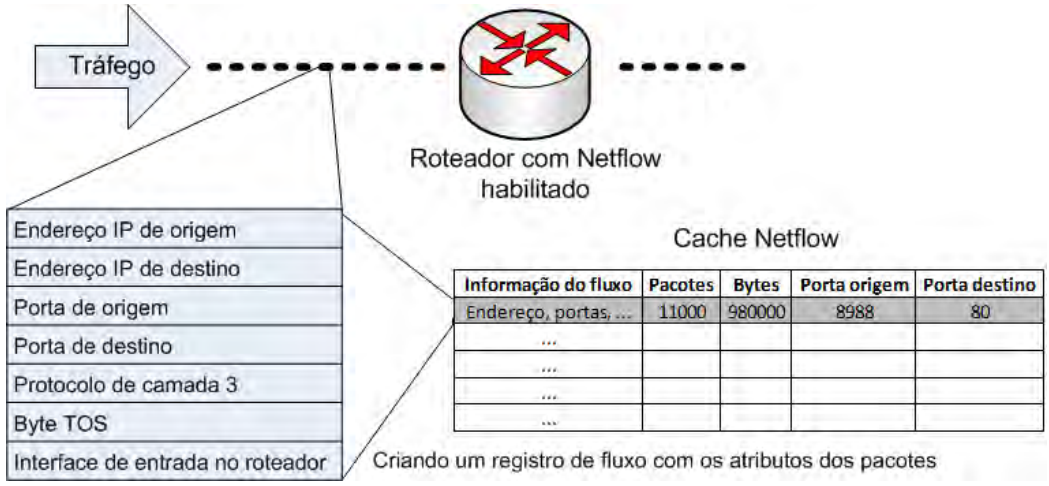

Figura 4. Esquema do padrão Cisco Netflow de fluxos de rede.
Figura adaptada de (Cisco, 2009).

Uma vez que os fluxos gerados nos equipamentos são exportados e armazenados, passam a compor uma fonte valiosa de informações. É possível obter detalhes de cada conexão estabelecida por qualquer máquina pertencente ao ambiente monitorado, fator este muito relevante na análise de segurança.

Historicamente, a análise de rede aparece relacionada à captura de pacotes para a detecção de eventos, como os ataques, e na grande maioria das vezes faz referência a biblioteca *libpcap* (Project, 2007). Esta biblioteca é mundialmente difundida no contexto da captura de pacotes sendo que muitas ferramentas utilizam-na. Há de ficar claro que a utilização de fluxos de dados Netflow como fonte de informação não é uma substituição à captura e análise de pacotes com *libpcap*. A utilização desta biblioteca é voltada para a análise de conteúdo dos pacotes como ocorre com o Snort (Roesch, 2009), ou outros sistemas de detecção de intrusos como o apresentado em (Bonifacio *et al.*, 1998).

Na busca de escalabilidade e da diminuição da carga computacional exigida para análise, os fluxos Netflow foram escolhidos como fonte de informações. Quando coletados de um roteador *stub* (*gateway* de uma instituição, por exemplo) permitem uma visão completa de todas as sessões e conexões do ambiente, sem a análise de conteúdo dos pacotes, não inserindo, portanto, nenhum tipo de latência devido ao desencapsulamento. Ainda, a exportação de fluxos Netflow também pode ser ativada

em *switches* ou mesmo com softwares ligados às redes, a fim de diminuir a carga de um roteador central. Todos estes exportadores enviam fluxos a um coletor central responsável pela análise dos dados.

2.5 Estado da arte na detecção de eventos por análise de fluxos

Desde antes da proposta do IPFIX pelo IETF, diversas pesquisas têm sido realizadas com o objetivo de se estabelecer novos modelos e ferramentas de monitoria de redes. Durante este período as pesquisas com fluxos se concentraram em três grandes focos: a visualização de fluxos, a detecção de anomalias e a identificação de eventos nos fluxos de um ambiente.

O primeiro deles trata da relação visual de administradores de rede com eventos que ocorrem em seus ambientes. O principal argumento defendido por estes autores é o de que a visualização gráfica de determinados comportamentos facilita ao administrador inferir determinadas informações. As primeiras ferramentas de coleta já demonstravam o interesse de se utilizar informações visuais na análise de fluxos. Um dos primeiros conjuntos de ferramentas para manipulação de fluxos, o *flow-tools* (Fullmer, 2001) operava conjuntamente com o utilitário *flowscan* (Dave, 2000), capaz de gerar diversos tipos de gráficos, separados por fluxos, *bytes*, pacotes, redes e protocolos. Embora sejam trabalhos estritamente técnicos, essas ferramentas alavancaram outras pesquisas por facilitarem a coleta e análise dos fluxos em si.

A partir de então, os trabalhos se concentraram em desenvolver mecanismos mais hábeis de visualização. Segundo Cristina Abad (Abad *et al.*, 2004), *"a visualização desperta a capacidade cognitiva humana facilitando a associação com eventos que estariam obscuros em meio a grande quantidade de informações que um SDI gera"*. Em seu trabalho é desenvolvida uma ferramenta denominada VisFlowConnect capaz de gerar padrões visuais diferentes de acordo com as características de conectividade dos *hosts*. De modo geral, os padrões são baseados na quantidade de endereços envolvidos tais como comunicações N para 1, 1 para N e 1 para 1. Esta taxonomia é bastante utilizada nos sistemas monitores de fluxos pois estas classificações se relacionam diretamente com diversos tipos de ataque, como as negativas de serviço distribuídas.

Na mesma linha, Jon Oberheide utiliza-se de uma ferramenta denominada Flamingo, própria para visualização de tráfegos Netflow, inserindo modelos de três

dimensões, relacionando grandezas tais como conjunto de endereços, conjunto de portas, quantidade de bytes, dentre outras presentes nos fluxos Netflow (Oberheide; Goff; Karir, 2006). Cada tipo de evento gera determinada figura no espaço tri-dimensional.

Trabalhos recentes sobre visualização começam a utilizar métodos diferentes para objetivos diferentes. Em (Taylor *et al.*, 2009) existem três paradigmas diferentes. O primeiro é baseado na codificação por cor de séries temporais com o objetivo de mostrar aspectos comportamentais de um *host* individualmente. O segundo é chamado conjunto de conexões e mostra as interações entre *host* individuais e grupos de *hosts* (da mesma forma que o trabalho anterior, 1 para N e N para 1) . O terceiro é chamado visualizador de NetBytes e permite um detalhamento por portas e do comportamento de *host* com base no volume de dados em um período de tempo.

O segundo grande foco de pesquisas utilizando fluxos refere-se aos mecanismos de detecção de anomalias. Este é, sem dúvida, o objetivo da maior parte dos trabalhos publicados na área. Isto se deve a escalabilidade que os fluxos trouxeram para análise de tráfego em determinados nichos, tais como *backbones* de Internet. Estes pontos caracterizam-se por tráfegos intensos e *links* de alta capacidade. Assim, a possibilidade de monitorar estes *links* utilizando informações mais detalhadas despertou grande interesse nas pesquisas. Diversos mecanismos têm sido propostos para detectar anomalias. Dentre eles pode-se citar o uso de sistemas inteligentes, lógica difusa, métodos estatísticos, algoritmos especializados, algoritmos distribuídos, máquina de estado finito, dentre vários outros.

No terceiro foco, pesquisas têm sido realizadas com o objetivo de realizar identificações de eventos em meio a todo o tráfego de um ambiente, utilizando para tanto apenas informações contidas em fluxos. Exemplo disto é o sistema BLINC (Karagiannis; Papagiannaki; Faloutsos, 2005). Este utiliza um modelo que observa e identifica padrões de comportamento de *hosts* considerando a camada de transporte. Utilizando apenas informações dos fluxos é um classificador capaz de identificar que determinado *host* está trafegando pacotes de determinada aplicação. A maior contribuição está em permitir esta associação *host*-serviço sem utilizar definições de portas. Ele identifica padrões de tráfego HTTP, FTP, SMTP e outros, sem definir em que portas operam (método de classificação chamado ‘*on the dark’*). Muitas das características utilizadas por este sistema, como a definição de taxonomias de tráfego 1

para N, N para 1 e 1 para 1 são utilizadas pelo processador de fluxos do sistema sendo proposto.

Uma evolução da técnica utilizada em BLINC pode ser observada em (Laurent *et al.*, 2006). Embora continue utilizando fluxos, os autores defendem que apenas os cinco primeiros pacotes de uma aplicação são realmente necessários para identificá-la. A metodologia é dividida em duas etapas: treinamento e detecção. No treinamento são analisados pacotes capturados de diferentes aplicações e qual a relação que se observa com os fluxos gerados. A partir de então, tenta-se estabelecer uma descrição no âmbito dos fluxos da aplicação a ser detectada. A grande vantagem, segundo os autores, é a possibilidade de detecção *'online'*, no início da comunicação de determinado protocolo.

Dentro deste último foco, são encontrados trabalhos semelhantes ao sistema proposto. No entanto, em nenhum deles existe a possibilidade de se criar assinaturas para a monitoria '*online'* de um ambiente de rede.

Em (Myung-Sup *et al.*, 2004) os autores demonstram como a agregação de informações em fluxos de dados contribuem para a diminuição do processamento de informações de segurança. O modelo proposto visa realizar uma detecção de alterações no padrão de tráfego utilizando algoritmos especializados no processamento das informações fornecidas pelos fluxos.

O objetivo é bastante semelhante ao módulo de detecção de anomalias do trabalho sendo desenvolvido: detectar tráfego anômalo causados por *worms*, ataques de negativa de serviço (DoS), ataques de negativa de serviço distribuído (DDoS) e varreduras (*scans*). Os autores defendem ainda que existem ataques que podem ser representados de maneira simples e outros que necessitam uma análise de todas as informações fornecidas pelos fluxos. Ainda, neste paradigma, é possível detectar ataques mutantes que alteram algum tipo de informação (como uma porta de destino, por exemplo) baseando-se no padrão de tráfego que este ataque gera entre a fonte e o destino. Embora não seja mencionado o conceito de assinaturas, na própria definição dos padrões de tráfego algumas características são descritivas de eventos. A Figura 5 e a Figura 6 mostram algumas das regras utilizadas para a detecção de eventos como *flooding* em protocolos ICMP (*Internet Control Message Protocol*), TCP e UDP, utilizando características como IP de origem e destino, portas de origem e de destino e número e tamanho dos pacotes.

L : Large S : Small

Attacks / Property	scanning		flooding				
	host	network	TCP SYN	smurf	fraggle	ping-pong	general (ICMP,UDP,TCP) flooding
flow count	L	L	L	L	L	S	L or S
flow size/flow	S	S	S	L or S	L or S	L	L or S
packet count/flow	S	S	S	L or S	L or S	L	L or S
packet size	S	S	S	L or S	L or S	L or S	L or S
total bandwidth	L or S	L or S	L or S	L	L	L	L
total packet count	L or S	L or S	L or S	L	L	L	L
property	1 destination	1 port		ICMP broadcast	UDP broadcast	reflecting port	

Figura 5. Regras utilizadas para a definição de padrões de tráfego.
Figura retirada de (Myung-Sup *et al.*, 2004).

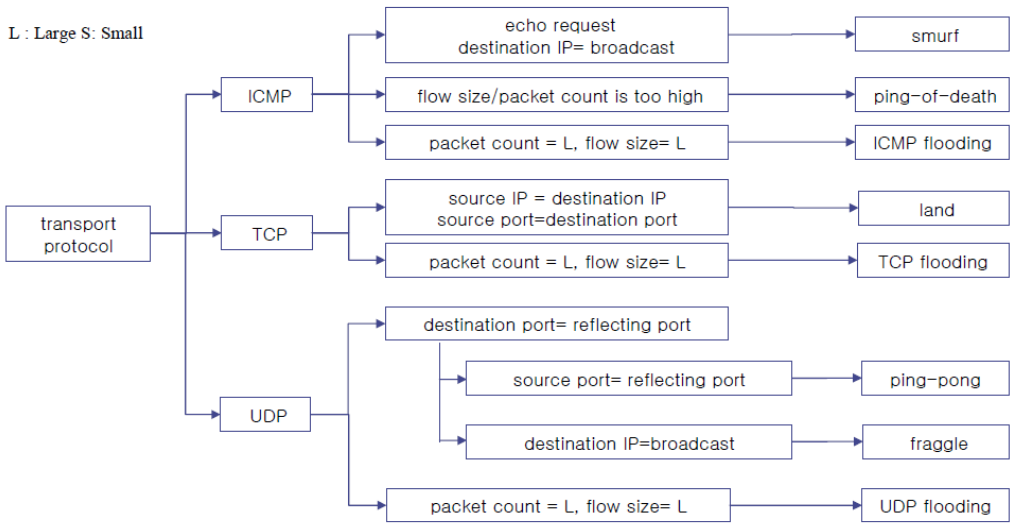

Figura 6. Seqüência de detecção para ataques comuns, utilizando fluxos de rede.
Figura retirada de (Myung-Sup *et al.*, 2004), modificada.

Outra semelhança é a arquitetura deste sistema, também baseada no padrão IPFIX de exportação e coleta de fluxos, como pode ser observado na Figura 7.

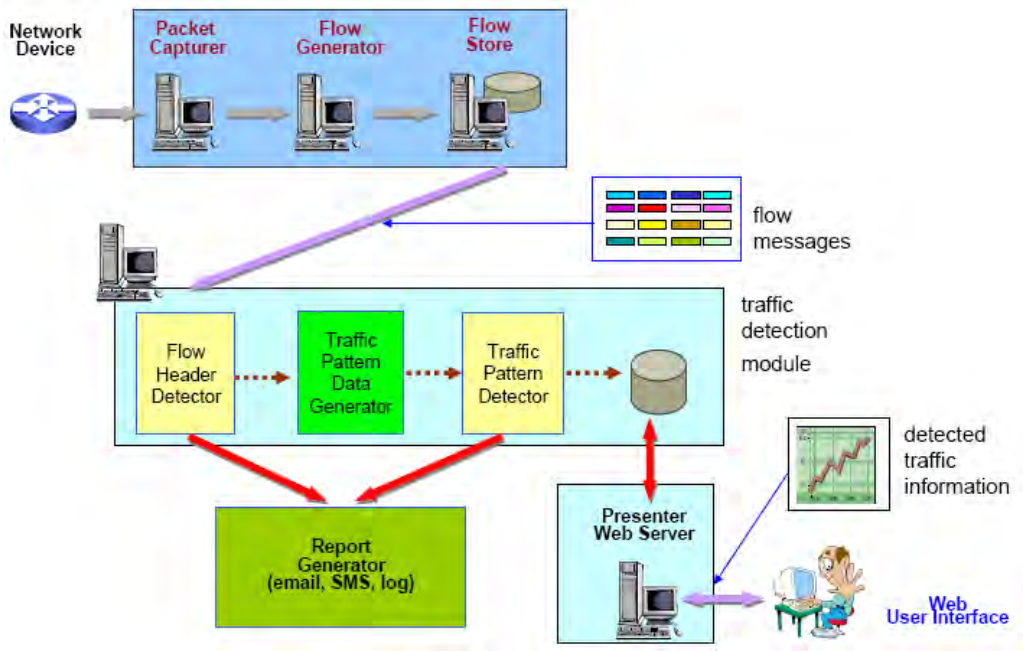

Figura 7. Arquitetura do sistema de detecção de anomalias de tráfego.
Figura retirada de (Myung-Sup *et al.*, 2004).

Dentre os diversos trabalhos analisados, a tentativa de utilizar assinaturas juntamente com fluxos de dados pode ser encontrada em (Dressler; Jaegers; German, 2007). O artigo trata de um trabalho não concluído que mostra a arquitetura utilizada, a metodologia e os primeiros resultados. Além da utilização do padrão IPFIX de arquitetura de fluxos, a utilização de bancos de dados relacional para o armazenamento destes aproxima este trabalho da metodologia utilizada neste projeto de pesquisa. A abordagem envolve a utilização de *honeypots* para captação de ataques e geração de 'assinaturas'. Os *honeypots* são sistemas de segurança desenvolvidos para serem comprometidos, atraindo atacantes e gerando diversas informações para o analista de segurança. Normalmente utilizam algum sistema de detecção de intrusos, como o Snort, bem como suas assinaturas. Na ocorrência de ataques, estes sistemas geram relatórios bastante completos, com as descrições do ataque, a assinatura utilizada em sua detecção, os fluxos de rede e a captura dos pacotes.

A partir desses alertas, (Dressler; Jaegers; German, 2007) geraram 'assinaturas' na forma de *scripts* que interagem com o banco de dados procurando a ocorrência do ataque ao *honeypot*, representado com as informações disponíveis nos fluxos. Um exemplo bastante interessante mostra a detecção do *worm* Dabber. O ataque do Dabber é caracterizado por conexões na porta 5554 com a transferência de 2000 a 6000 bytes. Em 5 segundos, conexões nas portas 443, 8967, 1023 e 9898 são abertas. A Figura 8 mostra um esquema da detecção do Dabber neste sistema.

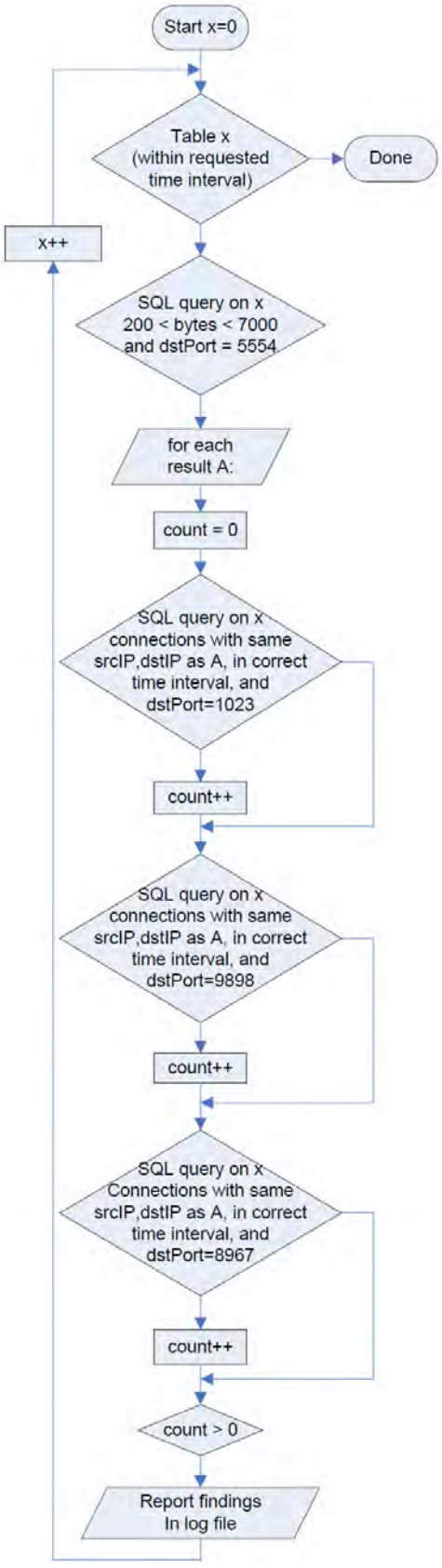

Figura 8. Detecção de atividades do *worm* Dabber em (Dressler; Jaegers; German, 2007).

O trabalho objetiva provar a viabilidade do mecanismo de detecção utilizando fluxos de dados. Não existe uma definição sólida de assinatura, sendo cada evento descrito por um conjunto de rotinas responsáveis por encontrar os passos de um ataque, sofrido pelos *honeypots*, em meio aos fluxos do ambiente. Os ataques analisados foram encontrados em meio aos registros de fluxos em um paradigma *post-mortem*. Não existe na realidade um processo de detecção, mas sim buscas pelas ocorrências dos eventos alertados pelo *honeypot* em meio aos fluxos. Em outras palavras, o processo desenvolve-se partindo dos *logs* do *honeypot* em busca dos fluxos que representem o evento em questão. Não há menção de uma tentativa de detecção *online*. Uma das propostas do modelo de rastreamento é realizar uma detecção o mais rápido possível, assim que os fluxos são recebidos. Embora o tempo para a exportação dos fluxos até o coletor e processador implique em uma análise *post-mortem*, utilizá-los o mais rápido possível resulta em grande eficiência nas operações de reação contra os eventos de segurança. O processo de detecção parte dos fluxos mais atuais recebidos em busca de assinaturas que os identifiquem como representativos de um evento na rede.

Trabalhos mais recentes propõem modelos semelhantes ao de rastreamento de fluxos. Em (Bin *et al.*, 2008) muitas características são semelhantes, como a utilização de banco de dados, JAVA, detecção de anomalias e descrição de eventos. Os autores utilizam várias metodologias para a detecção de anomalias como variância de similaridade, distância euclidiana e métodos estatísticos. Existe ainda um algoritmo detector de padrões que considera descrições de eventos utilizando dados dos fluxos. No entanto, não existe menção a um sistema de detecção *online*.

Outros trabalhos também apresentam modelos desenvolvidos para a análise descritiva de fluxos (Wang; Wang, 2008) (Chan; Shoniregun; Akmayeva, 2008). No entanto, nenhum deles menciona a detecção *online* de eventos nem define estruturas como as assinaturas utilizadas no sistema de rastreamento de fluxos. Apesar de possibilitar determinadas descrições, estas se limitam a consultas diretas nos fluxos. Por exemplo, uma descrição de evento é feita com base nas portas e na quantidade de bytes que determinado fluxo apresenta. Esses trabalhos não permitem que fluxos sejam relacionados entre si para compor uma descrição mais apurada de um evento, característica presente no sistema proposto neste texto. Alguns nem mesmo realizam agrupamentos para identificação de características de agrupamentos de endereços e portas, muito importantes na detecção de determinados eventos.

Este capítulo discutiu conceitos importantes para o entendimento deste trabalho, os eventos de maior importância verificados nas redes atuais e os mecanismos de segurança mais utilizados atualmente, buscando identificar suas vantagens e desvantagens. Foi abordada uma conceituação sobre a tecnologia de fluxos de rede, notadamente no padrão Netflow. Por fim, foi exposta uma discussão sobre a evolução das pesquisas com fluxos e quais os objetivos atuais nesta área.

O capítulo 3 discutirá todas as características do modelo de rastreamento de fluxos proposto neste trabalho, desde a arquitetura de armazenamento que utiliza álgebra relacional para a representação de agrupamentos de fluxos, até as características do sistema implementado, capaz de realizar detecções online de assinaturas geradas e cadastradas.

Capítulo 3 - O modelo de detecção de eventos baseado no rastreamento de fluxos

Este capítulo descreve o sistema de rastreamento de fluxos, bem como as metodologias utilizadas para seu desenvolvimento. Inicialmente é explicada a arquitetura de armazenamento do sistema, baseada em banco de dados relacional e na separação de dados entre memória e disco para fins de desempenho. Antes da discussão das assinaturas utilizadas pelo sistema, é mostrada uma discussão sobre o estado da arte na geração de assinaturas para eventos de segurança. Como será visto, diferentemente dos padrões e protocolos bem definidos nas redes de computadores, a geração de um padrão (*standard*) de assinaturas não é bem visto pela comunidade científica. Desta forma, o sistema proposto utiliza suas próprias assinaturas, com suas próprias características, tendo em vista a tendência da literatura em não padronizar este tipo de estrutura.

Em seguida, é descrita a utilização da álgebra relacional para a caracterização de tráfego e extração de informações sobre eventos de rede. Este conceito está relacionado com as assinaturas utilizadas pelo sistema. Então, são apresentados os tipos de assinatura utilizados, seus objetivos e a metodologia de geração dessas. Por fim, é apresentada uma visão geral do sistema produzido, suas funcionalidades atuais e futuras.

3.1 Arquitetura de armazenamento

O padrão IPFIX não estabelece um modelo de implementação dos sistemas de coleta e armazenamento de fluxos. Por exemplo, o *flow-tools* utiliza um padrão próprio de armazenamento, de forma que apenas utilizando suas bibliotecas é possível acessar os dados por ele armazenados. Uma das etapas deste projeto trata do desenvolvimento de uma aplicação responsável pela coleta e armazenamento de fluxos. A utilização de

banco de dados relacional fornece novos meios tanto para a detecção de anomalias quanto para ataques específicos, aumentando a eficiência do sistema.

A utilização de um banco de dados relacional para o armazenamento de fluxos insere uma imensa versatilidade na manipulação de informações sobre o tráfego de um ambiente. As facilidades proporcionadas pela Linguagem de Manipulação de Dados SQL contribuem fortemente para o acesso às informações de rede, principalmente para o caso de uma grande quantidade de dados, caso dos fluxos Netflow.

Alguns trabalhos têm sido realizados com o propósito de mostrar a eficiência da utilização de banco de dados no armazenamento de fluxos. (Bill, 2000) pode ser tomado como um dos trabalhos precursores na utilização de banco de dados em conjunto com fluxos. São expostas as vantagens que um sistema gerenciador de banco de dados pode inserir durante a manipulação de fluxos e como alguns eventos são facilmente detectados com simples consultas aos dados. Não existe, no entanto, preocupação com o desempenho do banco, otimizações de armazenamento, visto que são utilizados *clusters* e supercomputadores para o processamento dessas informações, gerando assim uma subutilização de processamento.

Neste contexto, um dos objetivos do sistema de coleta e armazenamento deste projeto é prover mecanismos de baixo custo, tanto financeiro quanto computacional (normalmente relacionados). Diversas otimizações em relação ao trabalho anteriormente citado foram realizadas, principalmente para o ganho de espaço em cada registro de fluxo e no que diz respeito às melhorias de desempenho para a análise *online*. É esperado que sistemas gerenciadores de banco de dados comuns possam ser utilizados para tarefas de monitoria, não requerendo grandes investimentos para a implantação.

Cada tabela de armazenamento do sistema é composta pelos campos mostrados na Tabela 1.

Tabela 1. Tamanhos dos campos utilizados nas tabelas de fluxos.

Campo	Tipo de dados	Representação	Tamanho
Srcaddr	Integer	sem sinal, não nulo	4 bytes
Dstaddr	Integer	sem sinal, não nulo	4 bytes
Input	Smallint	sem sinal, não nulo	2 bytes
Output	Smallint	sem sinal, não nulo	2 bytes
Packets	Integer	sem sinal, não nulo	4 bytes
Octets	Integer	sem sinal, não nulo	4 bytes
First	Timestamp	não nulo	4 bytes
Last	Timestamp	não nulo	4 bytes
Srcport	Smallint	sem sinal, não nulo	2 bytes
Dstport	Smallint	sem sinal, não nulo	2 bytes
tcp_flags	Tinyint	sem sinal, não nulo	1 byte
Prot	Tinyint	sem sinal, não nulo	1 byte
Tos	Tinyint	sem sinal, não nulo	1 byte

Cada registro de fluxo (entrada na tabela) consome 35 bytes, diferentemente dos 58 bytes utilizados no trabalho de Bill (Bill, 2000). Esta otimização se dá pela escolha dos tipos de dados necessários para a representação dos endereços além do descarte de alguns campos dos fluxos Netflow que não são necessários para a detecção de eventos.

O sistema gerenciador de banco de dados utilizado foi o MySQL (Microsystems, 2009). Trata-se de um SGDB de domínio público bastante utilizado em sistemas Linux. Os dados coletados atualmente somam cerca de 110 GB, armazenados em tabelas separadas por dia. Cada tabela diária possui em média 400 MB e cerca de 11 milhões de fluxos. As tabelas utilizam uma indexação com base em um campo de tempo de forma que as pesquisas em tabelas no disco são mais rápidas quando utilizando restrições de tempo.

Para aumentar a velocidade de verificação das assinaturas e possibilitar a análise *online* dos fluxos é utilizado um tipo de tabela denominado HEAP. Esta tabela é armazenada totalmente em memória tornando as consultas muito mais rápidas do que as realizadas em disco. Todos os fluxos recebidos pelo coletor são armazenados diretamente nesta tabela, chamada '*last30minutes*'. A cada 1 minuto, um procedimento é responsável por retirar os fluxos mais antigos da tabela em memória e armazená-los em sua respectiva tabela no disco. Este processo existe devido à restrição de exportação dos fluxos comentada anteriormente: é possível que um fluxo fique ativo no roteador sem ser exportado por até 30 minutos. Desta forma, mantendo os últimos 30 minutos em memória facilitam-se as consultas para detecção *online* e o processo de inserção na tabela diária, uma vez que esta operação é custosa devido à indexação por tempo. A Figura 9 mostra este esquema de operação da arquitetura de armazenamento.

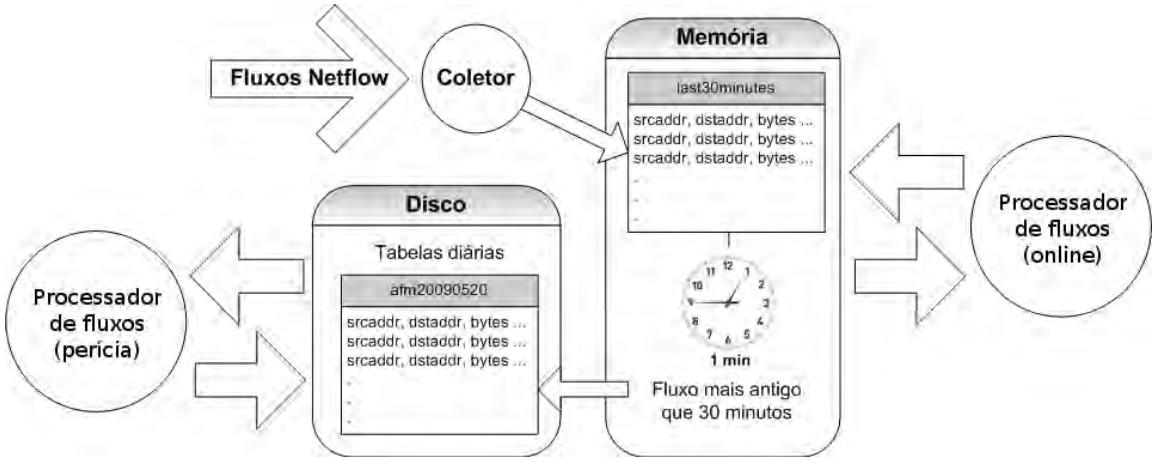

Figura 9. Arquitetura de armazenamento utilizando banco de dados relacional.

Esta arquitetura é a base de todo o sistema. Além de fornecer meios para que consultas rápidas sejam realizadas a fim de detectar eventos o mais rápido possível possibilita o armazenamento de informações de rede para que perícias *post-mortem* possam ser realizadas.

3.2 Utilizando a álgebra relacional para descrever classes de fluxos

Quando um fluxo é analisado é possível observar informações sobre como endereços e portas de origem e destino se comportam. A criação das assinaturas neste trabalho se relaciona fortemente com o modelo de armazenamento mencionado anteriormente. Os sistemas de banco de dados relacional utilizam a denominada álgebra relacional (Elmasri; Navathe, 2005) para representação dos dados e os operadores relacionais para sua manipulação. Estes conceitos nos permitem manipular e organizar dados de maneira que informações sobre o tráfego possam ser aferidas. Organizando informações segundo operações de agrupamento é possível definir uma taxonomia de tráfego que leva em consideração o número de *hosts* e portas comunicantes.

A seguir são apresentadas as definições e operadores da álgebra relacional que permitem a organização dos fluxos e a detecção de certas características. Para acompanhar as definições das operações serão utilizadas tabelas de fluxos que permitem uma melhor visualização de cada operação. Em todos os casos, uma tabela é chamada de “relação” e representada por um nome em maiúsculas. Cada relação é formada por “atributos”, ou em outras palavras, colunas. A tabela FLUXOS (Tabela 2) servirá de exemplo.

Tabela 2. Tabela FLUXOS: relação contendo dados sobre fluxos de um ambiente de rede.

Tabela FLUXOS							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	172.16.0.1	9000	6881	15	6000	2009-05-01 16:00:00	TCP
192.168.1.10	172.16.0.2	9001	6881	15	6000	2009-05-02 16:00:05	TCP
192.168.1.10	172.16.0.3	9002	6881	15	6000	2009-05-03 16:00:11	TCP
192.168.1.10	172.16.0.4	9003	6881	15	6000	2009-05-04 16:00:20	TCP
192.168.1.10	172.16.0.5	9004	6881	15	6000	2009-05-05 16:00:23	TCP
192.168.1.10	172.16.0.6	9005	6881	15	6000	2009-05-06 16:00:28	TCP
172.16.0.1	192.168.1.10	6881	9000	2	20	2009-05-01 16:00:03	TCP
172.16.0.2	192.168.1.10	6881	9001	2	20	2009-05-02 16:00:15	TCP
172.16.0.3	192.168.1.10	6881	9002	2	20	2009-05-03 16:00:15	TCP
172.16.0.4	192.168.1.10	6881	9003	2	20	2009-05-04 16:00:24	TCP
172.16.0.5	192.168.1.10	6881	9004	2	20	2009-05-05 16:00:33	TCP
172.16.0.6	192.168.1.10	6881	9005	2	20	2009-05-06 16:00:35	TCP
192.168.1.10	172.16.0.1	6881	8000	1	40	2009-05-01 16:04:00	UDP
192.168.1.10	172.16.0.2	6881	8001	2	50	2009-05-02 16:04:05	UDP
192.168.1.10	172.16.0.3	6881	8002	3	100	2009-05-03 16:04:11	UDP
192.168.1.10	172.16.0.4	6881	8003	4	150	2009-05-04 16:04:20	UDP
192.168.1.10	172.16.0.5	6881	8004	5	200	2009-05-05 16:04:23	UDP
192.168.1.10	172.16.0.6	6881	8005	6	250	2009-05-06 16:04:28	UDP
192.168.1.10	172.16.0.7	6881	8006	1	2000	2009-05-06 16:04:29	UDP
192.168.1.10	172.16.0.8	6881	8007	2	2000	2009-05-06 16:04:30	UDP
192.168.1.10	172.16.0.9	6881	8008	3	2000	2009-05-06 16:04:30	UDP
192.168.1.10	172.16.0.10	6881	8009	4	2000	2009-05-06 16:04:31	UDP
192.168.1.10	172.16.0.11	6881	8010	5	2000	2009-05-06 16:04:31	UDP

(a) ***seleção***, representada por σ, é utilizada para selecionar um subconjunto de tuplas que satisfaça uma condição booleana. Em geral a seleção é representada por

$$\sigma_{\text{<condição de seleção>}} (\text{RELAÇÃO}).$$

Por exemplo, a seleção SUBTAB_A(srcaddr, ..., prot) $\leftarrow \sigma_{(\text{dPkts} = 1)}$ (FLUXOS) gera como resultado a Tabela 3.

Tabela 3. SUBTAB_A: resultado de uma operação seleção.

SUBTAB_A							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	172.16.0.1	6881	8000	1	40	2009-05-01 16:04:00	UDP
192.168.1.10	172.16.0.7	6881	8006	1	2000	2009-05-06 16:04:29	UDP

(b) ***projeção***, representada por π, restringe os atributos de uma relação. Em outras palavras, seleciona colunas de uma relação. Em geral é representada por

$$\pi_{\text{<lista de atributos>}} (\text{RELAÇÃO}).$$

Por exemplo, a projeção SUBTAB_B(dstaddr) ← π (dstaddr) (FLUXOS) gera como resultado a Tabela 4.

Tabela 4. SUBTAB_B: resultado de uma operação projeção.

SUBTAB_B
dstaddr
172.16.0.1
172.16.0.2
172.16.0.3
172.16.0.4
172.16.0.5
172.16.0.6
192.168.1.10
192.168.1.10
192.168.1.10
192.168.1.10
192.168.1.10
192.168.1.10
172.16.0.1
172.16.0.2
172.16.0.3
172.16.0.4
172.16.0.5
172.16.0.6
172.16.0.7
172.16.0.8
172.16.0.9
172.16.0.10
172.16.0.11

(c) ***função agregada***, representada por A, indica o agrupamento de tuplas a partir de algum atributo, possibilitando a aplicação de outras funções independentes como *soma, média, máximo, mínimo, contar*, entre outras. Em geral é representada por

<atributos do agrupamento> A <lista de função> (RELAÇÃO).

Por exemplo, uma função agregada pode ser aplicada à tabela FLUXOS, utilizando a função independente CONTAR para os endereços destino, fixando o atributo endereço origem. Esta função agregada é representada por

SUBTAB_C(srcaddr, N_dstaddr) ← (srcaddr) A CONTAR(dstaddr) (FLUXOS),

produzindo como resultado a Tabela 5.

Tabela 5. SUBTAB_C: resultado de uma operação função agregada utilizando a função independente CONTAR.

SUBTAB_C	
srcaddr	**N_dstaddr**
192.168.1.10	17
172.16.0.1	1
172.16.0.2	1
172.16.0.3	1
172.16.0.4	1
172.16.0.5	1
172.16.0.6	1

(d) ***equijunção***, representada por $^{[X]}$, seleciona dados de duas relações (tabelas) de forma que algum(s) atributo(s) nestes dois conjuntos satisfaça(m) a condição de igualdade. Em geral é representada por

$$\text{RELAÇÃO1}^{[X]}{}_{\text{<atributo de junção>}}\ \text{RELAÇÃO2}.$$

Por exemplo, pode-se aplicar uma equijunção entre as relações SUBTAB_A e SUBTAB_B obtidas anteriormente, utilizando o atributo *dstaddr*. Esta operação é representada por SUBTAB_D(srcaddr, ..., prot) ← SUBTAB_A$^{[X]}{}_{\text{(dstaddr)}}$ SUBTAB_B.

O resultado, mostrado na Tabela 6, é uma relação contendo as tuplas de SUBTAB_A em que o atributo *dstaddr* aparece em igualdade com os valores de *dstaddr* na relação SUBTAB_B.

Tabela 6. SUBTAB_D: resultado de uma operação de equijunção entre as relações SUBTAB_A e SUBTAB_B, utilizando o atributo *dstaddr* para a junção.

SUBTAB_D							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	**172.16.0.1**	8000	6889	1	40	2009-05-01 16:04:00	UDP
192.168.1.10	**172.16.0.7**	8006	6889	1	300	2009-05-06 16:04:29	UDP

Neste caso, a relação resultante é a própria relação SUBTAB_A, uma vez que a relação SUBTAB_B contém todos os *dstaddr* encontrados em SUBTAB_A.

Em suma, considerando os atributos de uma tabela que armazena fluxos os endereços de origem e destino, portas de origem e destino, número de pacotes no fluxo, número de bytes, tempo de criação, tempo de finalização, *flags* do cabeçalho TCP e protocolo de camada de transporte, representados respectivamente por *srcaddr, dstaddr, srcport, dstport, dPkts, dOctets, first, last, tcp_flags* e *prot*, uma seleção será uma busca segundo alguma expressão de restrição, enquanto uma projeção será quais dos atributos serão mostrados no resultado. A função agregada é utilizada como uma forma de

restringir um atributo e aplicar uma função independente como *contar*, *soma*, *média*, *desvios*, *etc.*, no subconjunto das tuplas pertencentes à restrição. Por fim, a equijunção é utilizada para selecionar as ocorrências de dois subconjuntos de dados de forma que no mínimo um atributo esteja relacionado entre os dois subconjuntos. Por exemplo, uma equijunção de uma lista de IPs com uma tabela diária de fluxos resultará em todos os fluxos que possuem algum IP que consta na lista.

Pode-se definir então tráfegos 1 para N e N para 1, respectivamente, como:

$$SUBTAB1(srcaddr, n\_dstaddr) \leftarrow {}_{(srcaddr)}\mathcal{A}_{CONTAR(dstaddr)}(FLUXOS)$$

$$SUBTAB2(srcaddr, n\_dstaddr) \leftarrow \sigma_{(n\_dstaddr > 1)}(SUBTAB1)$$

$$1\_PARA\_N(srcaddr, dstaddr, srcport, dstport) \leftarrow \pi_{(srcaddr,dstaddr, srcport, dstport)}(SUBTAB2 \; {}^{[X]}_{(srcaddr)} FLUXOS)$$

$$SUBTAB1(dstaddr, n\_srcaddr) \leftarrow {}_{(dstaddr)}\mathcal{A}_{CONTAR(srcaddr)}(FLUXOS)$$

$$SUBTAB2(dstaddr, n\_srcaddr) \leftarrow \sigma_{(n\_srcaddr > 1)}(SUBTAB1)$$

$$N\_PARA\_1(srcaddr, dstaddr, srcport, dstport) \leftarrow \pi_{(srcaddr,dstaddr, srcport, dstport)}(SUBTAB2 \; {}^{[X]}_{(srcaddr)} FLUXOS)$$

Tráfego 1 para 1 são endereços que não participam dos agrupamentos 1 para N e N para 1. A partir desta taxonomia pode-se observar outros tipos de comportamentos para determinados protocolos a fim de descrevê-los no âmbito dos fluxos. Por exemplo, analisando fluxos de aplicações P2P (Peer-to-peer) de compartilhamento de arquivos pode-se observar duas etapas: controle e transferência. O controle é composto por pacotes UDP enquanto a transferência trata da conexão entre os pares para transmissão 'garantida' pelo TCP. Considerando primeiramente a etapa que utiliza o protocolo TCP, as características desta aplicação são:

Etapa 1: protocolo TCP, mais de 5 endereços destino, nos últimos 5 minutos, com portas origem e destino maiores que 1023, com média de pacotes maior que 10 e com média de bytes maior que 5000.

Etapa 2: protocolo UDP, *hosts* utilizando a mesma porta de origem e se comunicando com mais de 5 *hosts* diferentes, dentro dos últimos 15 minutos, com portas origem e destino maiores que 1023, número de pacotes entre 1 e 6 e número de bytes entre 40 e 400.

Dessa forma, pode-se representar uma assinatura para detecção de P2P como segue. A cada operação é mostrada a relação resultante, até a obtenção da relação final, com os dados do *host* que possui a aplicação P2P ativa.

Passo 1:

$$SUBTAB1(srcaddr, n\_dstaddr) \leftarrow {}_{(srcaddr)}\,\mathcal{A}_{CONTAR(dstaddr)}\ (FLUXOS)$$

Tabela 7. P2P: Passo 1 - SUBTAB1 - contar a quantidade de hosts destino para cada host origem.

Passo 1 - SUBTAB1	
srcaddr	**n_dstaddr**
192.168.1.10	17
172.16.0.1	1
172.16.0.2	1
172.16.0.3	1
172.16.0.4	1
172.16.0.5	1
172.16.0.6	1

$$SUBTAB2(srcaddr, n\_dstaddr) \leftarrow \sigma_{(n\_dstaddr > 5)}\ (SUBTAB1)$$

Tabela 8. Passo 1 - SUBTAB2 - selecionar apenas os que possuem mais de 5 destinos.

Passo 1 - SUBTAB2	
srcaddr	**n_dstaddr**
192.168.1.10	17

$$SUBTAB3(srcaddr,...,prot) \leftarrow \sigma_{(first = \text{últimos 5 min},\ srcport > 1023,\ dstport > 1023,\ prot = TCP)}\ (FLUXOS)$$

Tabela 9. Passo 1 - SUBTAB3 - selecionar apenas fluxos dos últimos 5 minutos, com portas maiores que 1023 e protocolo TCP.

Passo 1 - SUBTAB3							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	172.16.0.1	9000	6881	15	6000	2009-05-01 16:00:00	TCP
192.168.1.10	172.16.0.2	9001	6881	15	6000	2009-05-02 16:00:05	TCP
192.168.1.10	172.16.0.3	9002	6881	15	6000	2009-05-03 16:00:11	TCP
192.168.1.10	172.16.0.4	9003	6881	15	6000	2009-05-04 16:00:20	TCP
192.168.1.10	172.16.0.5	9004	6881	15	6000	2009-05-05 16:00:23	TCP
192.168.1.10	172.16.0.6	9005	6881	15	6000	2009-05-06 16:00:28	TCP
172.16.0.1	192.168.1.10	6881	9000	2	20	2009-05-01 16:00:03	TCP
172.16.0.2	192.168.1.10	6881	9001	2	20	2009-05-02 16:00:15	TCP
172.16.0.3	192.168.1.10	6881	9002	2	20	2009-05-03 16:00:15	TCP
172.16.0.4	192.168.1.10	6881	9003	2	20	2009-05-04 16:00:24	TCP
172.16.0.5	192.168.1.10	6881	9004	2	20	2009-05-05 16:00:33	TCP
172.16.0.6	192.168.1.10	6881	9005	2	20	2009-05-06 16:00:35	TCP

$$\text{SUBTAB4}(\text{srcaddr}, \text{média}(\text{dPkts}), \text{média}(\text{dOctets})) \leftarrow {}_{(\text{srcaddr})}\mathrm{A}_{\text{MÉDIA}(\text{dPkts}),\ \text{MÉDIA}(\text{dOctets})} \quad (\text{SUBTAB3})$$

Tabela 10. Passo 1 - SUBTAB4 - cálculo das médias de pacotes e bytes para cada *srcaddr* da relação SUBTAB3.

Passo 1 - SUBTAB4		
srcaddr	**MÉDIA(dPkts)**	**MÉDIA(dOctets)**
192.168.1.10	15	6000
172.16.0.1	2	20
172.16.0.2	2	20
172.16.0.3	2	20
172.16.0.4	2	20
172.16.0.5	2	20
172.16.0.6	2	20

$$\text{SUBTAB5}(\text{srcaddr}, \text{média}(\text{dPkts}), \text{média}(\text{dOctets})) \leftarrow \sigma_{(\text{MÉDIA}(\text{dPkts}) >= 10,\ \text{MÉDIA}(\text{dOctets}) >= 5000)} \quad (\text{SUBTAB4})$$

Tabela 11. Passo 1 - SUBTAB5 - seleção de entradas que possuem média de pacotes maior ou igual a 10 e média de bytes maior ou igual a 5000.

Passo 1 - SUBTAB5		
srcaddr	**MÉDIA(dPkts)**	**MÉDIA(dOctets)**
192.168.1.10	15	6000

$$\textbf{1\_PARA\_N\_PASSO1}(\text{srcaddr}) \leftarrow \pi_{(\text{srcaddr})} (\text{SUBTAB5})$$

Tabela 12. Passo 1 - 1_PARA_N_PASSO1 - a projeção resulta no atributo *srcaddr*, resultado do passo 1.

Passo 1 - 1_PARA_N_PASSO1
srcaddr
192.168.1.10

Passo 2:

$$\text{SUBTAB1}(\text{srcaddr, srcport, n\_dstaddr, n\_dstport}) \leftarrow {}_{(\text{srcaddr, srcport})}\,\mathsf{A}_{\text{CONTAR(dstaddr), CONTAR(dstport)}}\,(\text{FLUXOS})$$

Tabela 13. Passo 2 - SUBTAB1 - função agregada para contar a quantidade de endereços e portas destino, dada uma combinação de endereço e porta origem.

Passo 2 - SUBTAB1			
srcaddr	**srcport**	**n_dstaddr**	**n_dstport**
192.168.1.10	9000	1	1
192.168.1.10	9001	1	1
192.168.1.10	9002	1	1
192.168.1.10	9003	1	1
192.168.1.10	9004	1	1
192.168.1.10	9005	1	1
172.16.0.1	6881	1	1
172.16.0.2	6881	1	1
172.16.0.3	6881	1	1
172.16.0.4	6881	1	1
172.16.0.5	6881	1	1
172.16.0.6	6881	1	1
192.168.1.10	6881	11	11

$$\text{SUBTAB2}(\text{srcaddr, srcport, n\_dstaddr, n\_dstport}) \leftarrow \sigma_{(\text{n\_dstaddr} > 10,\ \text{n\_dstport} > 10)}\ (\text{SUBTAB1})$$

Tabela 14. Passo 2 – SUBTAB2 - seleção apenas das tuplas que possuem a quantidade de endereços e portas destino maiores que 10.

Passo 2 - SUBTAB2			
srcaddr	**srcport**	**n_dstaddr**	**n_dstport**
192.168.1.10	6881	11	11

$$SUBTAB3(srcaddr, ..., prot) \leftarrow FLUXOS\ ^{[X]}_{(srcaddr,\ srcport)}\ SUBTAB2$$

Tabela 15. Passo 2 – SUBTAB3 - equijunção das relações FLUXOS e SUBTAB2 resultando nas tuplas de FLUXOS que possuem os atributos *srcaddr* e *srcport* de iguais aos de SUBTAB2.

Passo 2 - SUBTAB3							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	172.16.0.1	6881	8000	1	40	2009-05-01 16:04:00	UDP
192.168.1.10	172.16.0.2	6881	8001	2	50	2009-05-02 16:04:05	UDP
192.168.1.10	172.16.0.3	6881	8002	3	100	2009-05-03 16:04:11	UDP
192.168.1.10	172.16.0.4	6881	8003	4	150	2009-05-04 16:04:20	UDP
192.168.1.10	172.16.0.5	6881	8004	5	200	2009-05-05 16:04:23	UDP
192.168.1.10	172.16.0.6	6881	8005	6	250	2009-05-06 16:04:28	UDP
192.168.1.10	172.16.0.7	6881	8006	1	300	2009-05-06 16:04:29	UDP
192.168.1.10	172.16.0.8	6881	8007	2	350	2009-05-06 16:04:30	UDP
192.168.1.10	172.16.0.9	6881	8008	3	400	2009-05-06 16:04:30	UDP
192.168.1.10	172.16.0.10	6881	8009	4	400	2009-05-06 16:04:31	UDP
192.168.1.10	172.16.0.11	6881	8010	5	400	2009-05-06 16:04:31	UDP

$$SUBTAB4(srcaddr, ..., prot) \leftarrow \sigma_{(first = \text{últimos 15 min},\ srcport > 1023,\ dstport > 1023,\ dPkts \text{ entre 1 e 6},\ dOctets \text{ entre 40 e 400},\ prot = UDP)}(SUBTAB3)$$

Tabela 16. Passo 2 – SUBTAB4 - seleção apenas das tuplas referentes aos últimos 15 minutos, com portas maiores que 1023, número de pacotes entre 1 e 6, número de bytes entre 40 e 400 e protocolo UDP.

Passo 2 - SUBTAB4							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	172.16.0.1	6881	8000	1	40	2009-05-01 16:04:00	UDP
192.168.1.10	172.16.0.2	6881	8001	2	50	2009-05-02 16:04:05	UDP
192.168.1.10	172.16.0.3	6881	8002	3	100	2009-05-03 16:04:11	UDP
192.168.1.10	172.16.0.4	6881	8003	4	150	2009-05-04 16:04:20	UDP
192.168.1.10	172.16.0.5	6881	8004	5	200	2009-05-05 16:04:23	UDP
192.168.1.10	172.16.0.6	6881	8005	6	250	2009-05-06 16:04:28	UDP

$$SUBTAB5(srcaddr, ..., prot) \leftarrow (SUBTAB4\ ^{[X]}_{(srcaddr)}\ \mathbf{1\_PARA\_N\_PASSO1})$$

Tabela 17. Passo 2 – SUBTAB5 - equijunção das relações SUBTAB4 e 1_PARA_N_PASSO1, resultando nos fluxos satisfazem as restrições do passo 1 e do passo 2.

Passo 2 - SUBTAB5							
srcaddr	**dstaddr**	**srcport**	**dstport**	**dPkts**	**dOctets**	**first**	**prot**
192.168.1.10	172.16.0.1	6881	8000	1	40	2009-05-01 16:04:00	UDP
192.168.1.10	172.16.0.2	6881	8001	2	50	2009-05-02 16:04:05	UDP
192.168.1.10	172.16.0.3	6881	8002	3	100	2009-05-03 16:04:11	UDP
192.168.1.10	172.16.0.4	6881	8003	4	150	2009-05-04 16:04:20	UDP
192.168.1.10	172.16.0.5	6881	8004	5	200	2009-05-05 16:04:23	UDP
192.168.1.10	172.16.0.6	6881	8005	6	250	2009-05-06 16:04:28	UDP

$$\mathbf{P2P}(\text{srcaddr}) \leftarrow \pi_{(\text{srcaddr})} (\text{SUBTAB5})$$

Tabela 18. Passo 2 - P2P - projeção do atributo *srcaddr*, mostrando o *host* envolvido nos dois passos descritores de atividade P2P.

Passo 2 - P2P
srcaddr
192.168.1.10

Os operadores da álgebra relacional, em conjunto com a arquitetura de armazenamento, possibilitam a criação de diversas assinaturas segundo esta mesma metodologia de passos.

3.3 O conceito de assinaturas

A tecnologia Netflow surgiu após exemplos reais de disseminação mundial de artefatos maliciosos como o CodeRed (Microsoft, 2001), Slammer (Microsoft, 2002) e MSBlast (Microsoft, 2003), sendo considerada uma das principais tecnologias na detecção de anomalias. O Netflow possibilita, em uma análise de rede, obter respostas sobre quem, o que, de onde, como e quando um tráfego ocorreu. No entanto, é uma tecnologia de fornecimento de informações para aplicações de rede e não define nenhuma maneira de utilização destas informações. Assim sendo, diversas metodologias podem ser criadas para as mais diversas finalidades relacionadas às redes de computadores.

Após o surgimento dos *worms*, e principalmente devido ao seu poder de disseminação automática que os tornam uma das maiores ameaças na Internet, suscitou a idéia de que seria possível descrevê-los de alguma forma em meio às pesquisas em

detecção de intrusos. Esta descrição, porém, não é baseada apenas na geração de assinaturas que identifiquem *payloads*, mas sim em muitas outras características. A definição de uma assinatura está relacionada com um conceito denominado 'linguagem de descrição de ataques'. Ataques contra sistemas computacionais podem ser considerados manifestações em termos de eventos, de diferentes naturezas (Vigna; Eckmann; Kemmerer, 2000), como pacotes em uma rede, chamadas de sistemas ou registros de *logs* gerados por uma aplicação. Segundo (Vigna; Eckmann; Kemmerer, 2000) as descrições de ataques são diferentes segundo seus escopos e naturezas de modo que seis classes de linguagem podem ser estabelecidas, sendo a linguagem de descrição de ataques apenas uma destas classes:

Linguagem de evento: são usadas para descrever eventos e normalmente possuem uma representação em linguagem natural. Exemplos desta linguagem são as utilizadas pelo *TCPDUMP* e sistemas de *log* em ambientes UNIX, como o *syslog*.

Linguagem de reação: são usadas para especificar uma ação de reação a ser tomada perante uma detecção de ataque. A maioria dos sistemas de detecção de intrusos não possuem uma linguagem de reação definida. Quando realizam tarefas reativas utilizam *scripts* e bibliotecas próprias. Uma operação que seria descrita por uma linguagem deste tipo seria a reprogramação de um *firewall*, por exemplo.

Linguagem de relatório: são responsáveis por reportarem a ocorrência de ataques aos administradores, trazendo informações sobre o caso em um nível de abstração superior. Realizam tarefa semelhante à linguagem de evento, mas em um nível de abstração diferente. Uma padronização de linguagens de relatório é vista como bastante benéfica para a integração de ferramentas de segurança. Exemplo disto é o padrão IDMEF (*Intrusion Detection Message Exchange Format*) (Debar; Curry; Feinstein, 2007) que passa por etapas de padronização no IETF (*Internet Engineering Task Force*).

Linguagem de correlação: são linguagens utilizadas para correlacionar alertas de vários sistemas na tentativa de reconhecer o 'evento final'. Elas especificam relações entre ataques para identificar tentativas coordenadas de intrusão.

Linguagem de exploração: são linguagens utilizadas para descrever os passos a serem seguidos para a realização de uma intrusão. Sem uma padronização iminente, normalmente são utilizadas linguagens que possibilitam execução, como C, C++, Perl, Shell, Python etc. Um exemplo clássico são os *exploits*, escritos nas mais diversas linguagens.

Linguagem de detecção: também conhecidas como ‘linguagens de ataques’, ou ‘linguagens de descrição de ataques’, são a base para os sistemas de detecção de intrusão. São linguagens que provêm mecanismos e abstrações a fim de tornar possível a identificação de manifestações de ataque. O exemplo mais próximo é a linguagem utilizada para definir regras no Snort.

Desta classificação pode-se perceber que a composição de uma assinatura abrange diversos escopos, de acordo com o que ela pretende representar. Uma assinatura será responsável por desempenhar determinado papel dentro de alguma das classes anteriores. Na prática, a utilização do termo assinatura está relacionada exclusivamente à linguagem de detecção. Por exemplo, embora existam outras linguagens no Snort, como a linguagem de eventos para emissão dos alertas e *logs*, referências às assinaturas do Snort são claramente entendidas como sendo as estruturas da linguagem de detecção.

Esta tendência de classificação é bastante promissora dentro da segurança. Por exemplo, uma padronização de linguagem de exploração simplificaria muito a geração de casos de teste para novos sistemas. O mesmo ocorreria com as linguagens de evento, de reação e de relatório. No entanto, os próprios autores desta classificação (Vigna; Eckmann; Kemmerer, 2000) sugerem que a padronização das linguagens de detecção e correlação é extremamente prejudicial para a comunidade de detecção de intrusos. Existe ainda um vasto campo a ser explorado nesta área e a definição de um padrão único de linguagem de detecção reprimiria a exploração de novas idéias. A exploração dos fluxos de dados Netflow é um exemplo disto. A definição de meios para detectar eventos utilizando fluxos está vinculada à utilização de uma nova linguagem de detecção, com características próprias dos fluxos.

Muitos trabalhos têm se baseado nas classificações anteriores objetivando o estabelecimento de linguagens para descrição de ataques, tanto para facilitar a criação quanto para aumentar a eficiência de assinaturas. Notadamente, estes trabalhos são mais voltados para o paradigma de abuso com detecção baseada tanto em *host* quanto em rede. STATL (Steven; Giovanni; Richard, 2002) é uma linguagem para especificação de assinaturas baseada em uma representação por diagramas de estado. Esta abordagem permite a descrição de uma penetração como uma seqüência de ações executadas por um atacante. É uma linguagem que pode ser utilizada em sistemas de detectores de intrusos restritos a detecção por abuso. ADeLe (Cédric; Mé, 2001) é outra linguagem de descrição de ataque cujo principal objetivo é combinar todo o conhecimento disponível

sobre um determinado ataque em uma descrição de alto nível. Permite tanto detecção baseada em *host* quanto baseada em rede. Uma característica interessante é que assinaturas na linguagem ADeLe podem ser utilizadas em diversos outros sistemas detectores de intrusos. Estas assinaturas são descrições em alto nível, podendo ser adaptadas para diversos sistemas. São estruturas significativas que podem ser entendidas como um conjunto das linguagens de exploração, detecção, correlação e reação. Outro trabalho importante é (Rubin; Jha; Miller, 2005) na qual uma grande contribuição é dada em relação a geração de assinaturas eficientes. Rubin et. al. propõem uma metodologia de construção de assinaturas baseada em duas linguagens formais: assinatura de sessão e invariante do ataque. A primeira linguagem descreve diversas características que refinam uma assinatura e auxiliam na diminuição de falso-positivos e falso-negativos. A segunda representa a parte invariante do ataque como um padrão de bytes, por exemplo, de maneira reduzida a fim de minimizar erros de detecção. O processador de assinaturas é uma máquina de estados finitos capaz de representar os estados de um ataque diante das descrições das assinaturas. Rubin et. al. ainda mostram as características que diferem a metodologia proposta das linguagens ADeLe e STATL.

Diferentemente das linguagens e abordagens até então discutidas, SHEDEL (Meier; Bischof; Holz, 2002) é uma linguagem cujo objetivo principal é prover meios para descrever uma assinatura sem se preocupar com sua detecção, isto é, preocupar-se com a descrição do ataque e não como ele será detectado. Para tanto, esta linguagem utiliza o conceito de *evento* como abstração central. Uma assinatura é a descrição de um ataque em sua totalidade, por meio de eventos (sub-eventos, passos e características do ataque) que se relacionam temporal e hierarquicamente. Estes princípios podem ser comparados com os necessários para a descrição de um ataque utilizando fluxos de dados. A representação de eventos, passo a passo, com suas características temporais e no modo como se relacionam estão presentes na tecnologia Netflow.

Por todas estas considerações sobre a geração de assinaturas pode-se dizer que a utilização de um padrão pela comunidade para descrever *ataques* não é bem visto. Cada linguagem e assinatura estão adequadas e relacionadas com a metodologia de detecção utilizada. As pesquisas recentes em detecção por abuso apresentam, em sua maioria, uma tendência a utilização de máquinas de estados que possibilitam uma representação temporal dos sucessivos passos de um ataque. Mesmo assim, não existe uma linguagem que possa ser utilizada para a geração de assinaturas de modo geral. O surgimento de

novas linguagens de descrição de ataques e conseqüentes assinaturas são provas da evolução das pesquisas no desenvolvimento de novas metodologias para a segurança de redes.

3.4 Tipos de assinatura

O sistema de rastreamento de fluxos utiliza dois tipos de assinaturas: de abuso e de anomalia. É importante mencionar que o termo ‘assinatura’ normalmente está relacionado com a análise de conteúdo, de forma que uma assinatura descreve uma seqüência de bytes dentro de um pacote. No sistema de rastreamento de fluxos, as assinaturas devem ser entendidas como descrições de eventos. Outro conceito que pode causar confusão é ‘assinatura de anomalia’. Normalmente, sistemas detectores de anomalias utilizam mecanismos que detectam comportamentos diferentes do considerado normal. De maneira breve, explicando com maiores detalhes adiante, o rastreador de fluxos pode utilizar diversos mecanismos para detecção de anomalias, bastando implementá-los dentro do sistema. A assinatura trata dos parâmetros para que este mecanismo seja utilizado, diferindo bastante das assinaturas comuns até então conhecidas dos SDIs. Na seção 3.4.2 será apresentada uma assinatura de anomalia simples com o objetivo de detectar discrepâncias em algumas variáveis do tráfego, tornando o conceito mais claro.

Ambas as assinaturas do sistema rastreador de fluxos são estruturas que utilizam informações presentes nos fluxos e são organizadas em *passos*. Cada passo corresponde a uma característica causada pelo evento a ser detectado. Por exemplo, se um evento realiza duas conexões para dois *hosts* destino diferentes, uma assinatura pode conter dois passos, responsáveis por detectar cada uma das conexões.

Dentre as informações presentes nos dois tipos de assinaturas estão:

Id: identificador único da assinatura (evento).

Passo: indica uma etapa da detecção do evento. Uma assinatura pode ter quantos passos forem necessários, sendo que para a detecção final do evento, todos os passos deverão ser detectados. A não detecção de algum passo implica na não detecção do evento, uma vez que parte dele está ausente.

Código de operação: é um valor único que define uma operação a ser realizada dentro do processador de fluxos. Por exemplo, o código 0 não define nenhum tipo especial de operação, apenas indica que deve ser realizada uma busca nos fluxos de

acordo com as restrições indicadas. Do código 1 em diante, várias operações podem ser definidas. Como já mencionado, o sistema detector de anomalias pode utilizar mecanismos diversos. O código de operação indica qual mecanismo deve ser utilizado. Por exemplo, o código 1 indica que seja executada uma comparação entre as médias de pacotes (bytes ou fluxos) em intervalos de tempo indicados, tentando detectar uma queda ou aumento maior que uma porcentagem indicada em um dos parâmetros da assinatura. Ao analisar um passo de uma assinatura, o processador de fluxos verifica qual metodologia utilizar de acordo com o código da operação. Se um mecanismo que utiliza sistemas inteligentes for implementado no sistema, uma assinatura poderá conter um passo que utilize este mecanismo para identificar uma anomalia. Em suma, quando determinada informação não pode ser auferida diretamente dos fluxos com uma consulta comum (operação 0), uma nova operação que o faça pode ser inserida no sistema mediante implementação do código correspondente.

Intervalo de tempo: compreende o intervalo de fluxos a ser analisado para o passo em questão, dentro de uma assinatura. Os valores possíveis variam dentro do intervalo dos últimos 30 minutos. Assim, é possível analisar os *últimos 5 minutos*, *10 minutos*, *15 minutos* e assim sucessivamente, além de intervalos como *de 30 a 25 minutos passados*, *25 a 20 minutos passados* e assim sucessivamente. Estas restrições de intervalos ocorrem na interface de cadastramento. Embora a discretização ocorra em intervalos de 5 minutos estes podem ser menores. O que é necessário é utilizar um período que contenha uma quantidade de fluxos suficiente para que agrupamentos possam resultar em informações representativas, tais como médias, ou a contagem de endereços destinos na definição de tráfego 1 para N e N para 1.

Intervalo de execução: compreende o intervalo de tempo entre duas buscas pela assinatura. Desta forma, é possível realizar uma ‘janela deslizante’ sobre o tempo corrente, não perdendo informações. Por exemplo, se um passo deve analisar os últimos 10 minutos de fluxos, o intervalo de execução pode ser escolhido como 5 minutos. Desta forma não haverá perda de informações. Supondo o passo executar aos 15 minutos de determinada hora, ele analisará de 5 a 15 minutos. Quando em 20 minutos da mesma hora, analisará de 10 a 20 minutos. Se o intervalo de execução for igual ao intervalo de tempo (campo anterior), para o mesmo caso, a assinatura analisaria de 5 a 15 minutos e de 15 a 25 minutos, não analisando o intervalo compreendido entre 10 e 20 minutos.

Características de tempo: são características extraídas da relação entre os atributos *first* (início) e *last* (término) dos fluxos. Por exemplo, em varreduras (*scans*) é comum um fluxo começar e terminar exatamente no mesmo segundo. Desta forma, é possível indicar restrições como *last – first menor que, last – first maior que, first wildcard, last wildcard,* em que *wildcard* pode ser um tempo expressado da forma "2008-12-01 12:%%:%%" (todos os fluxos das 12h).

Restrições de endereços: indicam a operação de *seleção*, onde uma restrição é imposta para a busca. Os endereços origem e destino podem ser restringidos com operadores *igual, diferente* ou *wildcard,* que pode representar um conjunto de endereços. Ainda, podem ocorrer *relações entre os endereços dos passos*. Por exemplo, na definição do passo 2 de um evento, pode-se restringir o endereço de destino como sendo igual ao endereço de origem do passo 1 (ou de qualquer outro passo já executado).

Restrições de portas: restrições dos atributos *srcport* e *dstport* para as buscas. Podem utilizar operadores como *igual, diferente, maior (igual), menor (igual), entre* e *se relacionar com as portas de um passo anterior*. Por exemplo, a porta origem do passo 2 deve ser igual a porta destino do passo 1.

Interfaces do roteador: indicam as interfaces de entrada ou saída de um fluxo no roteador. Estes são valores inteiros que auxiliam na determinação de onde partiu determinada conexão.

Número de pacotes e número de bytes: são informações sobre a quantidade de pacotes e bytes de cada fluxo, sendo extremamente importantes para a detecção de padrões de aplicações. Além de restrições simples como as comparações de igualdade, estes dois campos permitem a utilização de modificadores como *média, desvio padrão, variância, mínimo, máximo* e *soma*, sendo os resultados comparados utilizando os operadores de igualdade como nos atributos anteriores. É importante mencionar que agrupamentos de tráfego e de serviços são utilizados em conjunto com estes modificadores. Por exemplo, em um passo pode ser necessário determinar a média de pacotes para os *hosts* com tráfego 1 para N, de dentro para fora do ambiente. A média de pacotes será calculada para cada *host* que apresenta tráfego 1 para N, considerando todos os fluxos partindo deste *host* e que estejam de acordo com as outras restrições.

Flags: são as *flags* do cabeçalho TCP. A busca será restringida aos fluxos que apresentarem *apenas* a combinação de *flags* informada ou no mínimo a combinação de *flags* informada, dependendo da escolha no momento do cadastro da assinatura.

Protocolo: restrição quanto ao protocolo de camada de transporte, podendo ser ICMP (embora não seja transporte), UDP, TCP ou UDP e TCP.

Opções: o campo opções é utilizado para armazenar informações que não estão diretamente relacionadas com aquelas provenientes dos fluxos (endereços, portas, etc).

Além destas informações, tanto as assinaturas de abuso quanto as de anomalia possuem dados específicos, como será abordado nas seções seguintes.

A metodologia de detecção de passos também é comum aos dois tipos de assinatura. Todos os passos devem ser encontrados para o evento ser detectado. A Figura 10 mostra como o processador de fluxos opera.

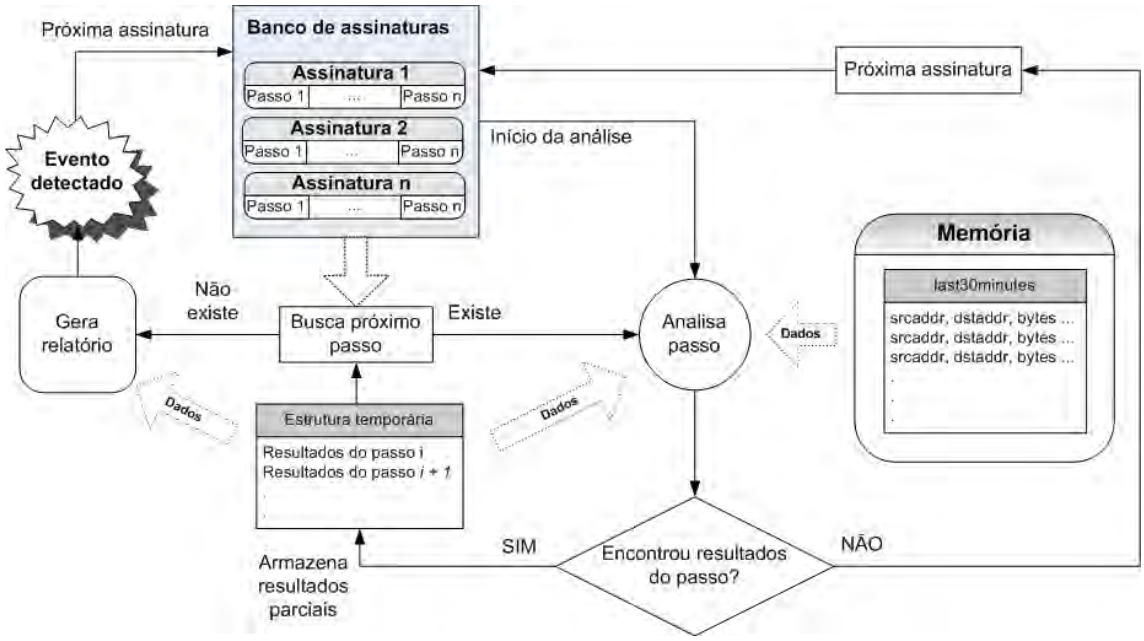

Figura 10. Fluxos de informações no processador de fluxos: interação com memória e detecção por passos.

Neste esquema toda a interação do processador com a fonte de fluxos é realizada com a memória, ou seja, representa a detecção *online* de eventos ocorrendo no ambiente. Existe também o caso da análise de eventos passados, em que dados armazenados em disco são confrontados com a base de assinaturas compreendendo uma análise pericial *post-morten*. Neste caso, o processo de detecção é mais lento devido à quantidade de dados analisada e ao tempo maior de obtenção de informações do disco quando comparado com dados em memória.

3.4.1 Assinaturas de abuso

Convencionou-se neste trabalho chamar assinaturas de abuso àquelas que descrevem fielmente um evento específico. Desta forma, eventos que geram comportamentos estáticos em uma rede são passíveis de detecção por este tipo de assinatura. Um evento estático é aquele que gera o mesmo comportamento (ou muito semelhante) em todas as suas instâncias. Essas assinaturas são utilizadas principalmente para detecção do tráfego de certas aplicações (normalmente que violam políticas de uso de redes) e para a identificação da disseminação de artefatos maliciosos, como os *worms*.

Uma vez que as assinaturas são organizadas em passos é importante ressaltar que para um evento ser detectado todos os passos devem ser encontrados nos fluxos de um ambiente. Se algum passo não for encontrado, o evento não será detectado. Este modo de operação pressupõe assinaturas diferentes para um mesmo evento, mas em ocasiões diferentes. Por exemplo, ao analisar um cliente P2P com compartilhamento de arquivos em um *host* sem restrição de tráfego nota-se um comportamento diferente de quando esta mesma aplicação está protegida por um *Firewall* que realiza filtragem de portas. Neste caso, duas assinaturas diferentes são necessárias para detectar o mesmo evento.

Dentre os campos próprios desta assinatura estão:

Tipo de tráfego: define agrupamentos de endereços como 1 para 1, N para 1 e 1 para N. Eventos N para N podem ser detectados pela intersecção de dois eventos 1 para N (dois passos), já que os fluxos são separados em entrada/saída.

Característica de serviços: são os agrupamentos de portas como *mesma porta de origem, mesma porta de destino* e *mesmas portas de origem e destino*. Esta informação, juntamente com o tipo de tráfego, define as operações de *função agregada*.

Atributos dos fluxos: podem ser selecionados quais atributos dos fluxos devem estar presentes no resultado do passo (*Srcaddr, Dstaddr, Input, Output, Dpkts, Doctets*, etc). Estas informações representam o operador *projeção*.

Contagem: as informações de contagem definem limiares para a detecção dos eventos. Podem ser contadas as *quantidades totais de fluxos* que atendem uma restrição, *de endereços origem e destino distintos* e a *quantidade de portas origem e destino distintas*. Para cada uma destas informações podem ser aplicados operadores como

igual, *diferente*, *maior (igual)*, *menor (igual)* e *entre*, seguidos do limiar. Estes limiares são definidos pelo administrador do sistema e são característicos de cada ambiente.

A Figura 11 mostra os campos necessários para o cadastramento de um passo de uma assinatura de abuso. Cada assinatura pode ser composta de quantos passos forem necessários, podendo ter ligações de valores entre cada um deles.

Figura 11. Campos necessários para o cadastramento de um passo de uma assinatura de abuso.

3.4.2 Assinaturas de anomalia

Diferentemente das assinaturas de abuso, as de anomalias são baseadas em limiares e não identificam aplicações ou eventos específicos. Anomalias são desvios de comportamento de alguma variável, como número de pacotes, bytes ou fluxos. Desta

forma, assinaturas de anomalia têm como objetivo detectar características não esperadas, utilizando diversas metodologias aplicáveis em variáveis como as anteriores.

O sistema atual utiliza um mecanismo baseado em médias para detecção de anomalias de fluxos, bytes ou pacotes. As assinaturas são criadas de forma a analisar o comportamento dos fluxos recém coletados e compará-lo com o comportamento considerado normal do ambiente. O comportamento normal é estabelecido pela análise anterior das médias das variáveis analisadas. Quando a média recente apresenta valores maiores ou menores que limiares definidos pelo administrador, uma anomalia é indicada.

Embora utilizadas apenas médias, diversos métodos podem ser aplicados bastando adicionar ao sistema o código (implementação) correspondente ao novo mecanismo. Trabalhos atuais mostram eficientes mecanismos baseados no isolamento de *outliers*, métodos estatísticos e sistemas inteligentes. Estes métodos podem ser inseridos no sistema, constituindo uma nova operação (código de operação).

Um exemplo de um evento detectável por uma assinatura de anomalia é mostrado na Figura 12. A assinatura consiste em verificar anomalias na média de fluxos por segundo, dentro dos últimos 5 minutos, comparando com valores de intervalos anteriores. Embora o evento não seja ‘identificado’, mas sim uma anomalia, trata-se de um ataque de DoS em um servidor web causado por SYN Flood.

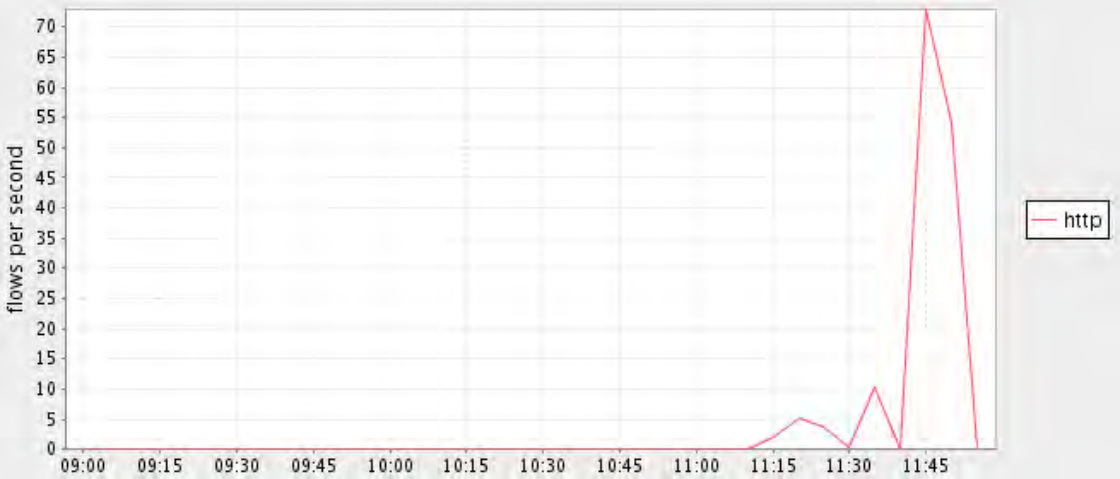

Figura 12. Visualização de um SYN Flood detectado por uma assinatura de anomalia.

A Figura 13 mostra os campos característicos de uma assinatura de anomalia. Esta pode ser composta de quantos passos forem necessários, podendo utilizar diferentes códigos de operação, um para cada passo.

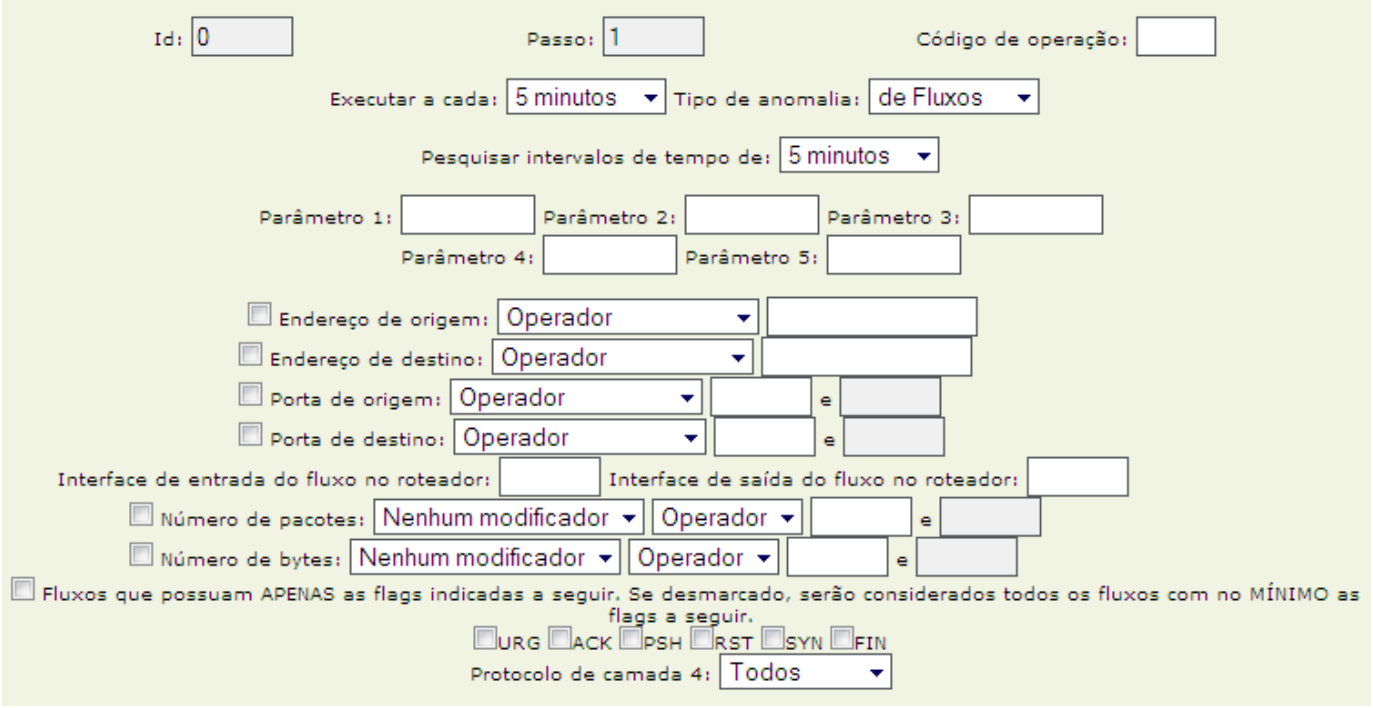

Figura 13. Campos de uma assinatura de anomalia.

Os campos característicos das assinaturas de anomalia são os 'Parâmetros'. Como anomalias são normalmente detectadas com base em limiares, existem cinco possíveis campos para o cadastramento destes limiares. O código que implementa o mecanismo de detecção definirá qual destes campos utilizar. Por exemplo, o código de operação 1 que implementa o mecanismo de médias reconhece como *Parâmetro 1* o número de intervalos anteriores utilizados para a determinação de um padrão normal. O *Parâmetro 2* é uma porcentagem que será utilizada para detectar aumentos ou quedas repentinas na média da variável sendo analisada (fluxos, pacotes ou bytes). A expressão utilizada por este código de operação, que uma vez satisfeita implica na detecção da anomalia, é

$$M_{atual} \leq \left(\frac{M_{acumulada} \times 2 + M_{intervalos}}{3}\right) \times \left(1 - \frac{P}{100}\right) \qquad (1)$$

ou

$$M_{atual} \geq \left(\frac{M_{acumulada} \times 2 + M_{intervalos}}{3}\right) \times \left(1 + \frac{P}{100}\right) \qquad (2)$$

onde:

M_{atual} representa a média de fluxos no intervalo sendo analisado *online*;

$M_{acumulada}$ representa a média dos valores de cada intervalo de tempo analisado, considerando todo o tempo em que o sistema executa;

$M_{intervalos}$ representa a média da variável analisada considerando a quantidade de intervalos indicada em *Parâmetro 1*;

P = uma porcentagem, indicada em *Parâmetro 2*, com $0 < P < 100$ para a expressão (1) e $P \geq 1$ para a expressão (2).

A utilização da média acumulada e da média de intervalos anteriores está relacionada ao comportamento dinâmico das variáveis monitoradas do ambiente (fluxos, pacotes ou bytes). Por exemplo, em determinado período da manhã a média de fluxos sofre certo aumento representando o início do expediente, quando os computadores são ligados e começam a gerar tráfego. Da mesma forma, ao entardecer, a média de fluxos decai em decorrência do término do expediente. Desta forma, a ponderação permite que a média atual (do momento sendo analisado) seja comparada com um valor ponderado que considera informações sobre o período do dia. A ponderação confere certa adaptabilidade no valor da média retratando como ela se comporta em determinados períodos do dia. Este mecanismo evita que haja uma generalização de valores, o que acarretaria falso-positivos.

3.5 Metodologia de geração das assinaturas

A geração das assinaturas depende da coleta de fluxos da aplicação para o qual ela será gerada. Uma vez que a assinatura é composta de passos, cada um desses compreende determinado comportamento da aplicação em relação ao tráfego gerado. Gerar uma assinatura para o sistema de rastreamento de fluxos é detectar padrões que se repetem todas as vezes que a aplicação em questão é executada.

A metodologia utilizada consiste em analisar isoladamente cada evento. Um *host* hospedeiro da aplicação é conectado a um *hub*. Neste dispositivo também é conectado um *host* gerador e coletor de fluxos que executa o programa *fprobe* (Astashonok, 2009). O *fprobe* é um software que utiliza o modo promíscuo de uma interface de rede gerando fluxos Netflow de todos os pacotes que consegue detectar no meio. Utilizando então um *hub* é possível gerar fluxos de todo o tráfego da máquina hospedeira de testes, coletando

nela mesma, utilizando a arquitetura de armazenamento já discutida. A Figura 14 mostra o ambiente utilizado.

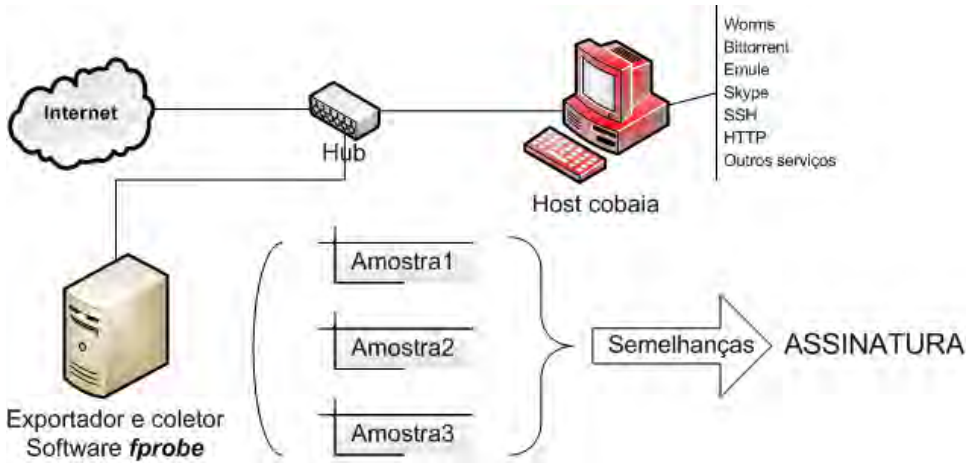

Figura 14. Ambiente de coleta de dados para geração de assinaturas.

Estabelecido o ambiente de coleta, para cada aplicação a ser descrita por uma assinatura foram geradas três amostras de fluxos, cuidando para que apenas pacotes desta aplicação fossem gerados. Então, comportamentos comuns entre as três foram procurados. O número de três amostras foi considerado como suficiente para definir que, se determinado padrão fosse encontrado nas três, este realmente era característico da aplicação sendo analisada. Todos os agrupamentos de fluxos, características temporais, de portas de comunicação, quantidade de bytes, de pacotes e de fluxos são consideradas. Determinadas as características presentes em todas as amostras, a assinatura é cadastrada no sistema.

3.6 Implementação do sistema

Todo o sistema de rastreamento de fluxos foi desenvolvido em linguagem Java. Além da padronização devido ao coletor também ser desenvolvido em Java, esta linguagem oferece a vertente Web de programação, o JSP. Assim, todo o processamento dos fluxos e acesso a banco são baseados em Servlets Java. A interface com o administrador/usuário é baseado em JSP, fornecendo assim a possibilidade de cadastramento de assinaturas, gerenciamento do sistema e visualização de relatórios.

O sistema gerenciador de banco de dados utilizado é o MySQL, de domínio público. Algumas funções internas deste sistema são utilizadas para facilitar a manipulação de dados, como ocorre com os atributos endereços de origem e destino.

Todos os valores armazenados foram detalhadamente estudados para não ocuparem espaço desnecessário. Os IPs, por exemplo, são armazenados como inteiros de 4 *bytes* e manipulados com funções próprias do MySQL.

Embora o sistema hospedeiro do sistema seja atualmente um Linux, tanto o Java quanto o MySQL possuem portabilidade para serem instalados e executados em diversos outros sistemas operacionais. Desta forma, o sistema de rastreamento de fluxos não se prende a um único sistema operacional.

3.6.1 Visão geral do sistema

Por tudo apresentado até então, é possível definir uma visão geral do sistema como na Figura 15. Nela estão presentes os componentes necessários para a tarefa de detecção e identificação de eventos em rede. Todo este processo inicia-se pela exportação de fluxos Netflow pelo roteador responsável pelo tráfego do ambiente monitorado. Os fluxos são recebidos pelo coletor, um software responsável por extrair as informações de cada fluxo e armazená-las em um formato próprio, de acordo com a arquitetura de armazenamento discutida neste capítulo. Estes fluxos passam determinado período de tempo em memória para posteriormente serem armazenados em disco. O processador de fluxos é o núcleo do sistema sendo responsável pela detecção dos eventos nos fluxos recentemente recebidos, utilizando para tanto informações contidas na base de assinaturas. Uma interface web possibilita ao administrador todo tipo de interação com o sistema, desde configurações até a visualização de resultados produzidos.

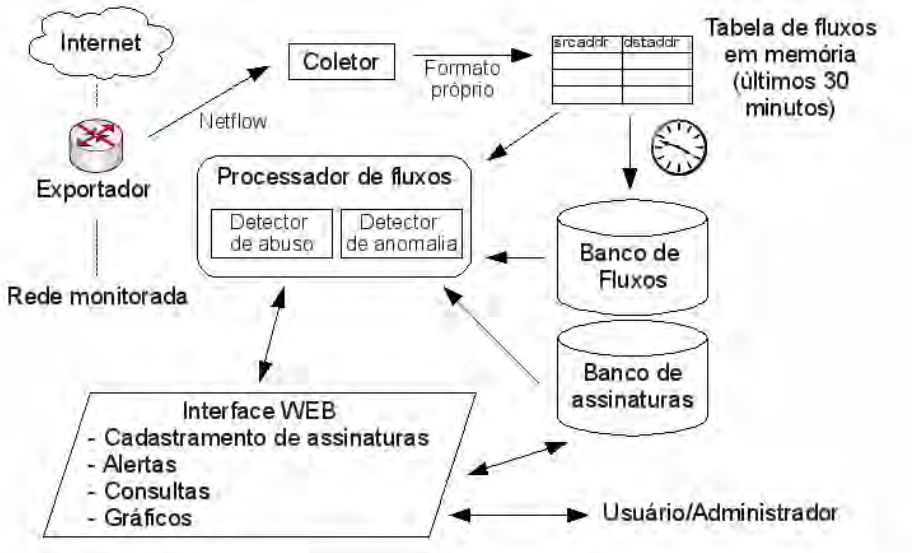

Figura 15. Visão geral do processo de detecção de um evento: etapas do sistema de rastreamento de fluxos.

3.6.2 Controles do sistema

A Figura 16 mostra a tela inicial do sistema. Os controles são divididos em três partes: assinaturas de abuso, assinaturas de anomalia e utilitário. Nas partes referentes às assinaturas, o administrador pode verificar os *Parâmetros* do banco de dados, acessando informações como a quantidade de assinaturas armazenadas em cada uma das bases. Os controles *Mostrar*, *Inserir* e *Remover* assinaturas permitem a manipulação completa destas estruturas dentro do sistema. Uma vez cadastrada, uma assinatura passa a fazer parte do conjunto de testes junto aos fluxos do ambiente monitorado, podendo ser imediatamente detectada.

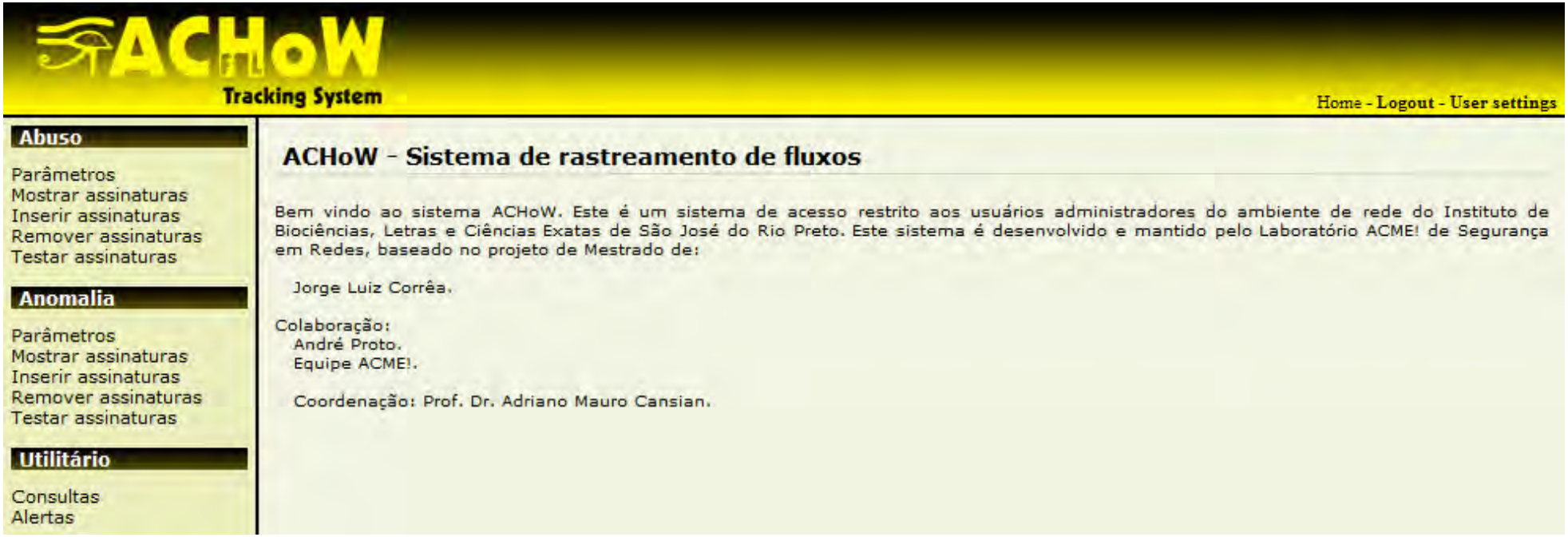

Figura 16. Tela inicial do sistema.

O controle *Testar assinaturas* permite que um conjunto de fluxos específico seja testado contra a base de assinaturas do sistema. Este conjunto pode ser relativo a dias de fluxos armazenados (tabelas diárias) ou a eventos externos (de outro ambiente) inseridos no banco de dados. Quanto aos dias, o sistema pode analisar tabelas de dias anteriores em busca de eventos de interesses, caracterizando uma ferramenta pericial. No caso dos eventos, um conjunto de fluxos coletado em outro ambiente pode ser inserido no banco de dados do sistema, em uma tabela específica do evento, para então ser analisado a fim de se detectar eventos neste conjunto. Por exemplo, se determinada instituição suspeita da ocorrência de determinado evento em sua rede, pode inserir o conjunto de fluxos no sistema e analisá-lo. Caso a suspeita seja de algum evento cuja assinatura ainda não tenha sido gerada ou cadastrada, esta tarefa deve ser realizada para que o sistema passe a procurar por este novo evento. Tanto para a análise diária quanto para a de eventos, é possível restringir o intervalo de tempo. Este controle, para as assinaturas de abuso, é mostrado na Figura 17. Para as assinaturas de anomalia o sistema é semelhante, apenas alterando o tipo de assinatura.

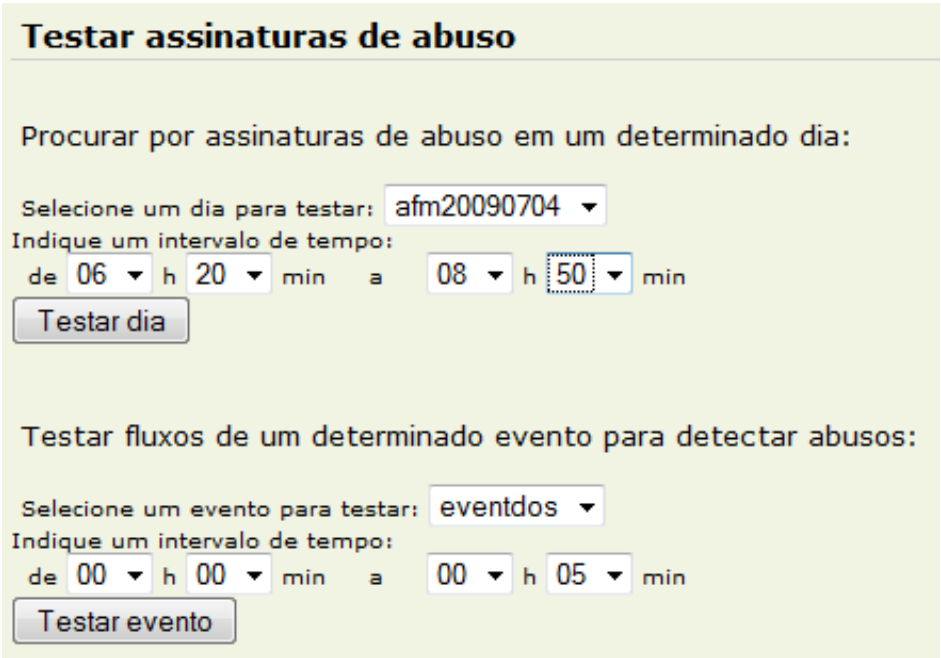

Figura 17. Controles para testar assinaturas de abuso em dias anteriores ou eventos específicos.

Por fim, o sistema possui na parte Utilitário os controles *Consultas* e *Alertas*. Em *Consultas* o administrador pode manipular facilmente os fluxos do ambiente realizando pesquisas rápidas utilizando todas as restrições possíveis da álgebra relacional. Desta forma, no caso de determinada suspeita sobre algum tráfego específico, o administrador pode consultar os fluxos do ambiente para constatar sua ocorrência ou não, imediatamente. Este controle permite consultar tanto dias armazenados quanto os fluxos atuais do ambiente de rede. A Figura 18 mostra a interface para este controle.

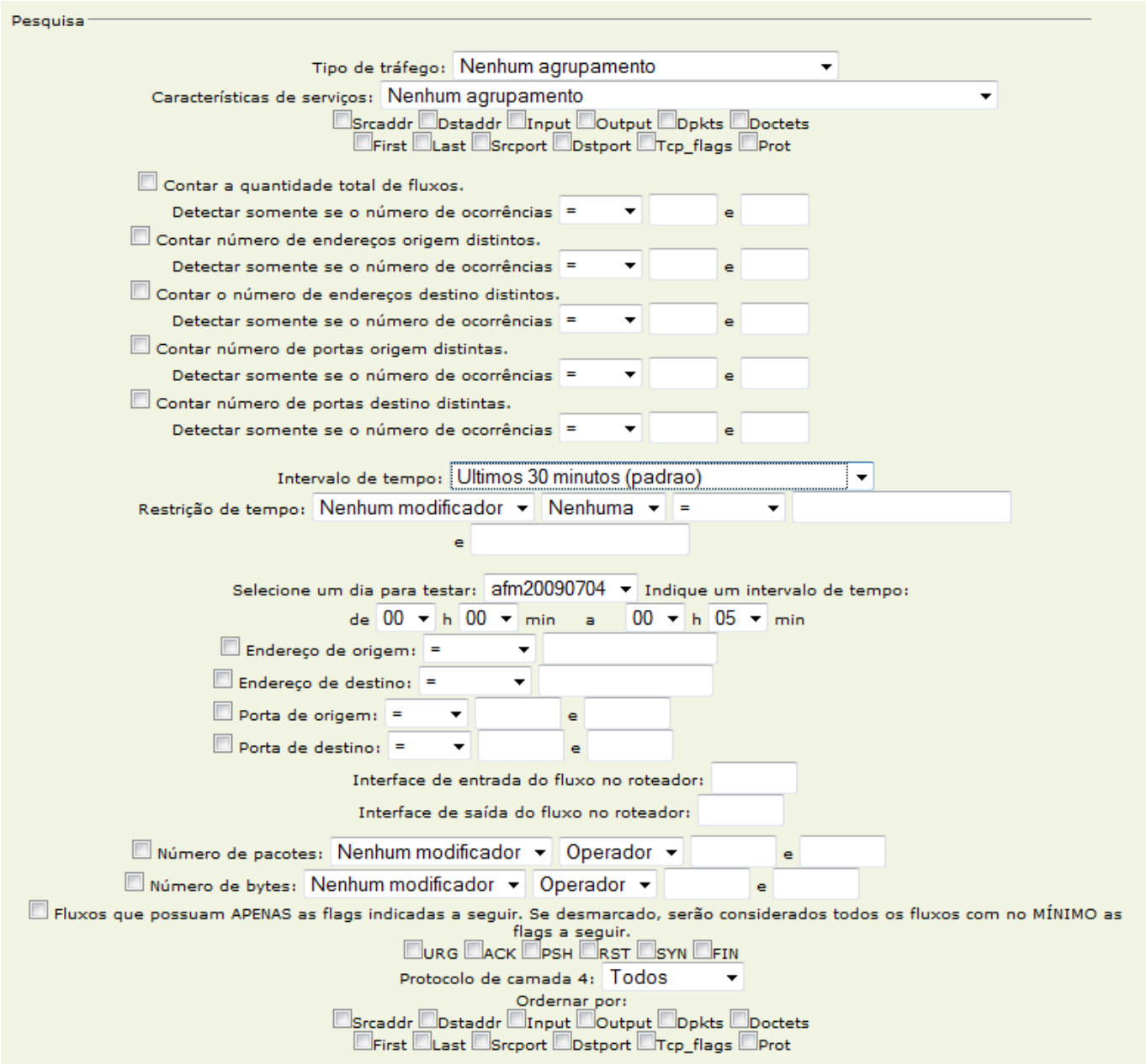

Figura 18. Controle *Consultas* possibilita a pesquisa de fluxos de maneira versátil e imediata.

O controle *Alertas* é onde o administrador pode observar os eventos sendo detectados no ambiente. São exibidos tantos os eventos detectados por assinaturas de abuso quanto por assinaturas de anomalia. Para cada resultado mostrando um *host* envolvido no evento o administrador pode clicar e obter informações relativas ao serviço de *whois* (Registro.Br, 2009) do endereço em questão, tais como o nome do responsável pelo endereço, nome da empresa ou instituição, contato, entre outras. A Figura 19 mostra alguns eventos identificados no controle *Alertas*.

Figura 19. Controle *Alertas* mostrando hosts envolvidos em eventos descritos por assinaturas de abuso.

Estes alertas são de eventos programados e não destrutivos dentro da rede monitorada.

O capítulo 3 descreveu o modelo de rastreamento de fluxos, passando por todas as etapas desde o armazenamento dos dados até a detecção de visualização dos relatórios de eventos. Desta forma, foi dada uma visão geral da metodologia utilizada no modelo e como esta foi implementada, dando origem ao sistema de monitoramento de fluxos denominado ACHoW.

O capítulo 4 mostrará o resultado de testes quantitativos com o modelo. Serão abordados os mecanismos utilizados para testes, os ambientes e as taxas de acertos e erros que mostram os níveis de eficiência obtidos.

Capítulo 4 - Resultados

Este capítulo apresenta os resultados desta pesquisa em termos de desempenho do sistema, assinaturas geradas e taxas de acertos e erros de classificação.

4.1 Considerações quanto ao desempenho do sistema

Conforme abordado no capítulo 1 deste trabalho, o desempenho das soluções de monitoria de redes passou a ser uma preocupação atualmente. A análise de desempenho do sistema implementado pode ser dividida em duas vertentes: *desempenho do sistema de coleta e armazenamento* e *desempenho do sistema processador de fluxos*. Antes da discussão sobre o desempenho convém tornar claras características do ambiente de testes. O sistema foi testado no ambiente de rede do Instituto de Biociências, Letras e Ciências Exatas de São José do Rio Preto (IBILCE – UNESP). Este ambiente conta com uma faixa de 4335 endereços IPs. Ao analisar o comportamento dos fluxos, é possível verificar uma utilização média simultânea de 600 IPs ativos. A banda do Instituto é de 34 Mbps. Todos os fluxos do ambiente são gerados e exportados no roteador principal do campus, compreendendo a representação de todo o tráfego da rede.

O desempenho do sistema de armazenamento está relativamente ligado à capacidade do sistema gerenciador de banco de dados, no caso, o MySQL. O ponto de maior interesse e que pode impactar na eficiência do sistema está na capacidade do SGBD (Sistema Gerenciador de Banco de Dados) de realizar inserções em um curto período de tempo. Esta característica é necessária pois a taxa de geração de fluxos é elevada, sendo proporcional à velocidade de comunicação do enlace analisado. Na rede mencionada, há momentos de pico com a geração de 4000 fluxos por segundo.

A utilização do MySQL sem nenhuma modificação no próprio SGBD ou em outra metodologias de manipulação de dados não permite uma taxa de inserção como esta. Assim, um dos objetivos da arquitetura de armazenamento desenvolvida é justamente possibilitar a manipulação de taxas elevadas como a mencionada. O primeiro fator que possibilita contornar esta restrição é a utilização de um *buffer* no coletor de

fluxos. Este *buffer* permite que os dados dos registros de fluxos sejam extraídos dos pacotes recebidos do roteador exportador e mantidos em memória até a inserção na tabela *last30minutes*. Desta forma, os fluxos não são inseridos diretamente em tabelas em disco. A própria tabela *last30minutes*, do tipo especial HEAP, é o segundo fator que auxilia na superação da restrição de taxa de inserção. As taxas de inserção dos SGBD são normalmente obtidas considerando inserções diretamente em disco. A utilização de uma tabela em memória tanto possibilita uma inserção mais rápida quando atua como um segundo *buffer*, quando a inserção ocorrer de fato em uma tabela em disco. Garante-se assim que, ao serem retirados da tabela em memória e armazenados em disco, não ocorrerá perda de fluxos e tal operação utilizará a taxa máxima que o SGBD pode atingir, caso seja um caso de pico.

Outra questão de desempenho sanada pela utilização da tabela em memória refere-se à indexação das tabelas em disco. A indexação de uma tabela é um arquivo contendo referências sobre quais blocos do disco contêm determinados dados. No caso das tabelas de fluxos, a indexação ocorre pelo campo *first*, ou seja, pelo tempo cronológico de criação de um fluxo. Assim, dado determinado tempo (horário) o SGBD é capaz de mapear em que bloco de disco inicia os dados referentes a este horário, não sendo necessária uma busca linear em toda a tabela. Para este tipo de indexação, a inserção fora de ordem caracteriza o pior caso, pois o índice deve ser refeito a cada inserção. Uma vez que é possível ocorrer a chegada de fluxos fora de ordem, a tabela em memória evita que este caso ocorra. Além disso, possibilita a inserção no melhor caso, visto que os fluxos são retirados da memória e inseridos no disco, sempre em ordem, eliminando, portanto, a latência de inserção devido à indexação.

Ainda, o fator de taxa de inserção não se limita apenas ao desempenho do SGBD. As linguagens de programação, como o JAVA utilizado no sistema, utilizam *drivers* para se conectarem ao SGBD. Estes *drivers* também inserem atrasos de inserção. *Benchmarks* realizados com o *driver* nativo do JAVA mostram que este pode inserir a uma taxa de 300 registros por segundo. No entanto, toda a arquitetura desenvolvida contorna este problema utilizando as metodologias mencionadas. Atualmente, é possível encontrar para um mesmo segundo, mais de 4000 registros inseridos nas tabelas em disco, superando as limitações do SBGB, indexação e *driver*.

Quanto ao desempenho do processador de fluxos, testes foram realizados para a detecção *online* de eventos e para a detecção em fluxos armazenados, como por exemplo, analisar um dia passado com as assinaturas cadastradas.

Na detecção *online* dos eventos a exigência de desempenho é dependente do número de *hosts* ativos na rede. O sistema foi testado por mais de 30 dias analisando todos os fluxos do ambiente mencionado. As assinaturas testadas serão discutidas detalhadamente em seções seguintes. Embora a rede analisada possua cerca de 600 IPs ativos simultaneamente, as classificações ocorreram sempre em menos de 2 minutos. Logo, foi possível garantir que a análise de um intervalo de fluxos terminasse em tempo hábil, ou seja, a análise ocorresse em um tempo menor que o próprio período analisado. No caso dos testes, as assinaturas foram geradas para analisar um período mínimo de 5 minutos de fluxos. Garantiu-se então que a cada 5 minutos todas as assinaturas eram verificadas. Os testes foram executados em um PC de configurações básicas, com processador de 3.2 GHz, 2 GB de memória RAM e 250 GB de disco.

Do ponto de vista da análise de fluxos armazenados, analisando as tabelas diárias armazenadas durante o período de testes foram obtidos os resultados mostrados na Tabela 19.

Tabela 19. Resultados quanto ao desempenho da análise de fluxos armazenados em disco, considerando uma tabela diária.

Média de fluxos por tabela	10.000.000 fluxos por tabela
Tamanho médio da tabela	354 MB
Duração média da análise (dia todo, 24 horas)	23 minutos

Estes resultados foram considerados satisfatórios dentro das características do ambiente monitorado. É possível ainda que algumas melhorias sejam realizadas de modo a obter tempos menores para as análises de fluxos armazenados. Nos testes realizados, estas análises utilizavam o mesmo sistema do coletor e analisador *online*. Logo, estes tempos referem-se à execução de tais tarefas concomitantemente aos testes das assinaturas para os fluxos que estavam sendo recebidos do ambiente. Uma arquitetura que divida as funções de análise *online* e análise pericial pode melhorar o desempenho do sistema, sem elevar consideravelmente os custos.

4.2 Assinaturas geradas

A geração de assinaturas é um processo demasiadamente lento. Encontrar as semelhanças entre as três amostras coletadas é, atualmente, um processo manual. Por este motivo, fazer testes como os realizados em sistemas de detecção de intrusos

convencionais torna-se uma tarefa dispendiosa, devido à inviabilidade de se gerar uma grande quantidade de assinaturas.

Existem atualmente alguns conjuntos de dados para testes disponíveis na Internet para pesquisadores da comunidade de detecção de intrusos. No entanto, estes conjuntos são mais voltados para sistemas que analisam conteúdos de pacotes ou utilizam metodologias com sistemas inteligentes. Como discutido, as assinaturas de fluxos empregam uma linguagem própria, não sendo possível a utilização destas informações para testar a capacidade de reconhecimento do sistema. Além disso, estes conjuntos normalmente trazem eventos antigos que não refletem nem se encaixam no novo paradigma de tráfego de rede discutido no capítulo 1.

A fim de auferir resultados quantitativos algumas assinaturas foram geradas. O principal objetivo destes testes é comprovar a funcionalidade do modelo proposto, utilizando uma metodologia de detecção por passos, auxiliada pela representação de classes de agrupamentos de fluxos.

4.2.1 *Scans* ou varreduras

Scans são varreduras realizadas por atacantes para obter informações de um ambiente de rede. As assinaturas de varreduras podem detectar tanto varreduras em rede (tentativa de determinar quais são os *hosts* ativos em uma rede) quanto varreduras em *host* (tentativa de determinar serviços ativos em um *host*).

A metodologia desta assinatura consiste em detectar uma grande quantidade de fluxos apenas com a *flag* SYN habilitada. Isto indica que algum *host* tentou abrir uma conexão, sendo que no caso de varreduras são tentativas sem sucesso. Como uma conexão gera dois fluxos, um para cada sentido, o conjunto de fluxos do sentido atacante-alvo terá fluxos apenas com a *flag* SYN. A Figura 20 mostra uma assinatura no sistema para detectar varreduras em um *host*. Dentre as restrições pode-se ver um tráfego 1 para 1, com no mínimo 50 portas de destino diferentes, tempo de duração do fluxo igual a zero, a *flag* SYN habilitada e considerando todos os protocolos. A restrição de endereço de destino visa analisar as varreduras apenas quando o destino se encontra dentro da rede do Instituto. No entanto, também é possível criar uma assinatura em que esta restrição aparecerá no endereço de origem, ocasião esta que detectará *hosts* de dentro do Instituto realizando varreduras.

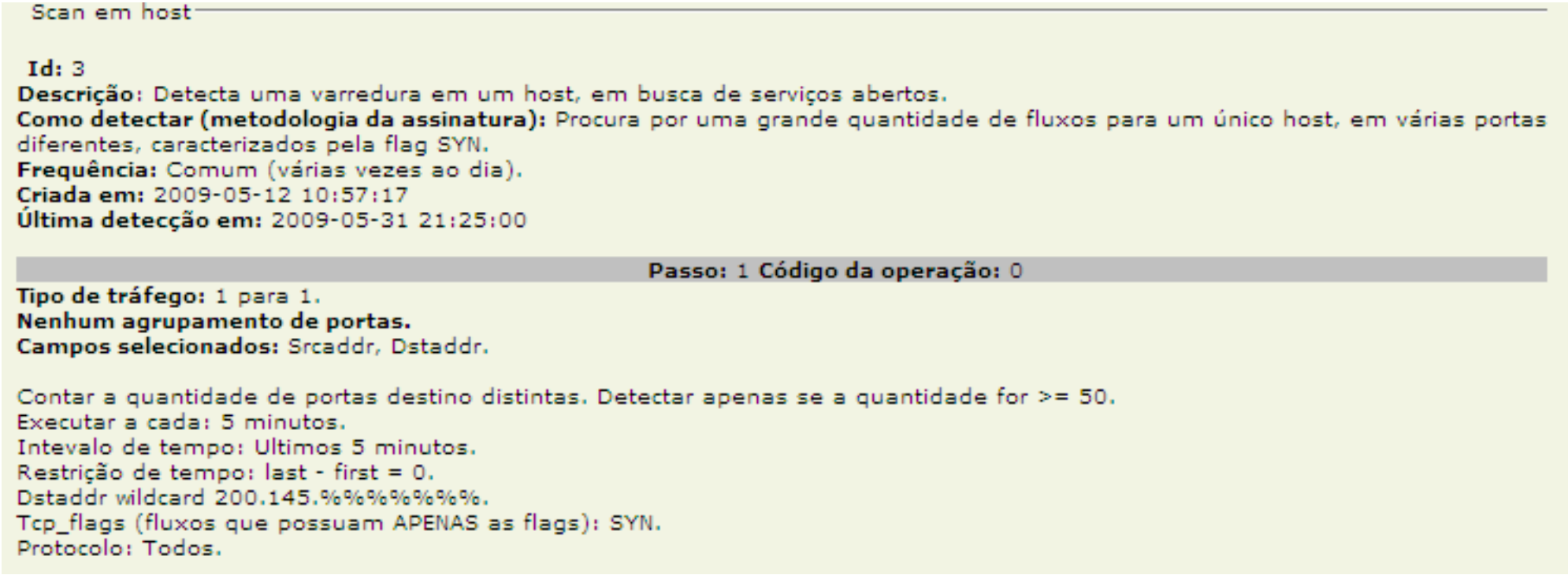

Figura 20. Assinatura para detecção de varreduras em *hosts*.

4.2.2 P2P

P2P é a nomenclatura utilizada para denominar aplicações de compartilhamento de arquivos tais como Emule, aMule, dentre outras. Estas aplicações se conectam em redes distribuídas como Gnutella, eDonkey e FastTrack. Estas aplicações geram problemas tais como o uso indevido de banda além de violações de políticas de rede e direitos autorais em arquivos protegidos.

Para a detecção destas aplicações foram geradas duas assinaturas devido ao comportamento distinto em dois casos diferentes. O primeiro caso é quando a aplicação executa em um *host* sem filtragem de portas. O segundo é quando este *host* está protegido por um sistema de *firewall* (com tradução de endereços, processo conhecido como NAT). Analisando as amostras coletadas, é possível perceber a diferença entre os casos e gerar as duas assinaturas.

Nos dois casos, o tráfego gerado é do tipo 1 para N a partir do *host* que executa o cliente P2P. Normalmente este tráfego envolve mais de 150 destinos. Os protocolos utilizados são o UDP e o TCP, sendo o primeiro para troca de informações sobre a rede distribuída e o segundo para a transferência dos arquivos. A Figura 21 mostra a assinatura para a detecção de P2P sem filtragem de portas.

P2P

Id: 5
Descrição: Atividade de compartilhamento de arquivos com redes do tipo Kazaa, Emule, FastTrack, entre outras, também conhecidas como aplicações P2P
Como detectar (metodologia da assinatura): Uma aplicação P2P gera dois padrões de fluxos. O primeiro consiste do startup e mensagens de gerenciamento, todas UDP. O segundo são conexões TCP que representam a transferência de arquivos entre os hosts da rede distribuída. Primeiramente verifica-se se houve o padrão de startup e em seguida detecta-se as características da transferência de arquivos
Frequência: Comum (várias vezes ao dia).
Criada em: 2009-04-14 11:33:47
Última detecção em: 2009-05-31 07:25:00

Passo: 1 **Código da operação:** 0

Tipo de tráfego 1 para N.
Característica de serviço: Mesma porta de origem.
Campos selecionados: Srcaddr, Srcport.

Contar a quantidade de endereços destino distintos. Detectar apenas se a quantidade for >= 150.
Contar a quantidade de portas destino distintas. Detectar apenas se a quantidade for >= 150.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: last - first <= 6.
Srcaddr wildcard 200.145.%%%%%%%%.
Protocolo: UDP.

Passo: 2 **Código da operação:** 0

Tipo de tráfego 1 para N.
Característica de serviço: Mesma porta de origem.
Campos selecionados: Srcaddr, Srcport.

Contar a quantidade total de fluxos. Detectar apenas se a quantidade for >= 5.
Contar a quantidade de endereços destino distintos. Detectar apenas se a quantidade for >= 5.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: last - first >= 50.
Srcaddr = srcaddr do passo 1.
Srcport = srcport do passo 1.
Protocolo: TCP.

Figura 21. Assinatura para detectar aplicações P2P sem filtragem de portas.

Para o caso com filtragem de portas, um passo é inserido para detectar uma repetição de fluxos com uma quantidade de pacotes e bytes idêntica. Estes fluxos referem-se a pacotes filtrados pelo *Firewall*, ocorrendo em apenas um sentido.

4.2.3 Bittorrent

Este protocolo é utilizado atualmente para troca de arquivos entre *hosts* distribuídos, mas com operações iniciais centralizadas em sistemas conhecidos como *trackers*. Um cliente normalmente estabelece uma conexão inicial com o *tracker* e obtém informações sobre outros *hosts* que possuem o mesmo arquivo. A partir de então, o cliente passa a gerar tráfego 1 para N a fim de obter partes do arquivo.

A geração da assinatura de Bittorrent foi a mais complexa dentre todas geradas, devido a quantidade de detalhes necessários para descrever este evento. Ela é composta de 7 passos que envolvem as seguintes características:

- Tráfego 1 para N, normalmente com mais de 20 *hosts* destino;
- O cliente utiliza várias portas origem diferentes para a troca de arquivos;
- As conexões possuem várias portas destino diferentes;
- Repetição no número de bytes e pacotes nos fluxos enviados pelo cliente;
- O cliente repete a utilização de alguma porta de origem (esta porta é configurada no cliente e utilizada para que outros *hosts* entrem em contato para troca de controles).

Para cada uma destas características foram definidos limiares considerados mínimos para que a aplicação seja detectada. A Figura 22 mostra a assinatura de Bittorrent com os limiares utilizados em cada um dos passos.

Bittorrent

Id: 2
Descrição: Atividade de compartilhamento de arquivos baseada no padrão Torrent
Como detectar (metodologia da assinatura): Detecta-se várias características do padrão como tráfego 1-N, repetições de portas origem e destino e repetições da quantidade de pacotes e bytes em cada fluxo
Frequência: Comum (várias vezes ao dia).
Criada em: 2009-04-02 11:44:33
Última detecção em: 2009-07-14 15:50:00

Passo: 1 **Código da operação:** 0

Tipo de tráfego 1 para N.
Nenhum agrupamento de portas.
Campos selecionados: Srcaddr.

Contar a quantidade de endereços destino distintos. Detectar apenas se a quantidade for >= 20.
Contar a quantidade de portas origem distintas. Detectar apenas se a quantidade for >= 45.
Contar a quantidade de portas destino distintas. Detectar apenas se a quantidade for >= 25.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: Srcaddr wildcard 200.145.%%%%%%%%.
Protocolo: TCP.

Passo: 2 **Código da operação:** 0

Nenhum agrupamento de tráfego.
Nenhum agrupamento de portas.
Campos selecionados: Srcaddr.

Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: Srcaddr = srcaddr do passo 1.
Bytes / Pacotes repetindo >= 25.
Protocolo: TCP.

Passo: 3 **Código da operação:** 0

Tipo de tráfego 1 para N.
Característica de serviço: Mesma porta de origem.
Campos selecionados: Srcaddr, Srcport.

Contar a quantidade total de fluxos. Detectar apenas se a quantidade for >= 4.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: Srcaddr = srcaddr do passo 2.
Protocolo: TCP.

Passo: 4 **Código da operação:** 0

Tipo de tráfego 1 para N.
Característica de serviço: Mesma porta de destino.
Campos selecionados: Srcaddr, Dstport.

Contar a quantidade total de fluxos. Detectar apenas se a quantidade for >= 4.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: Srcaddr = srcaddr do passo 3.
Protocolo: TCP.

Passo: 5 **Código da operação:** 0

Tipo de tráfego: N para 1.
Nenhum agrupamento de portas.
Campos selecionados: Dstaddr.

Contar a quantidade de endereços origem distintos. Detectar apenas se a quantidade for >= 20.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: Dstaddr = srcaddr do passo 4.
Protocolo: TCP.

Passo: 6 **Código da operação:** 0

Tipo de tráfego: N para 1.
Característica de serviço: Mesma porta de destino.
Campos selecionados: Dstaddr, Dstport.

Contar a quantidade total de fluxos. Detectar apenas se a quantidade for >= 10.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: Dstaddr = dstaddr do passo 5.
Protocolo: TCP.

Passo: 7 **Código da operação:** 0

Nenhum agrupamento de tráfego.
Nenhum agrupamento de portas.
Campos selecionados: Dstaddr.

Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: last - first = 0.
Dstaddr = dstaddr do passo 6.
Bytes / Pacotes repetindo >= 4.
Protocolo: TCP.

Figura 22. Assinatura de Bittorrent.

4.2.4 Ataques no serviço de SSH

Detecta ataques de força bruta ou dicionário, em que vários usuários e senha são testados continuamente em *hosts* que possuem o serviço de SSH (*Secure Shell protocol*). Uma vez que usuários e senhas são normalmente cadeia de caracteres com tamanhos parecidos, os fluxos gerados em cada tentativa apresentam certo padrão de comportamento quanto aos campos que tratam da quantidade de bytes e pacotes. A assinatura considera um mínimo de 15 tentativas, com duração de cada fluxo entre 5 e 12 segundos, média de pacotes entre 7 e 17, a quantidade de bytes por pacote entre 90 e 100 bytes, protocolo TCP e a *flag* SYN habilitada. A Figura 23 mostra a assinatura de detecção de ataques no SSH.

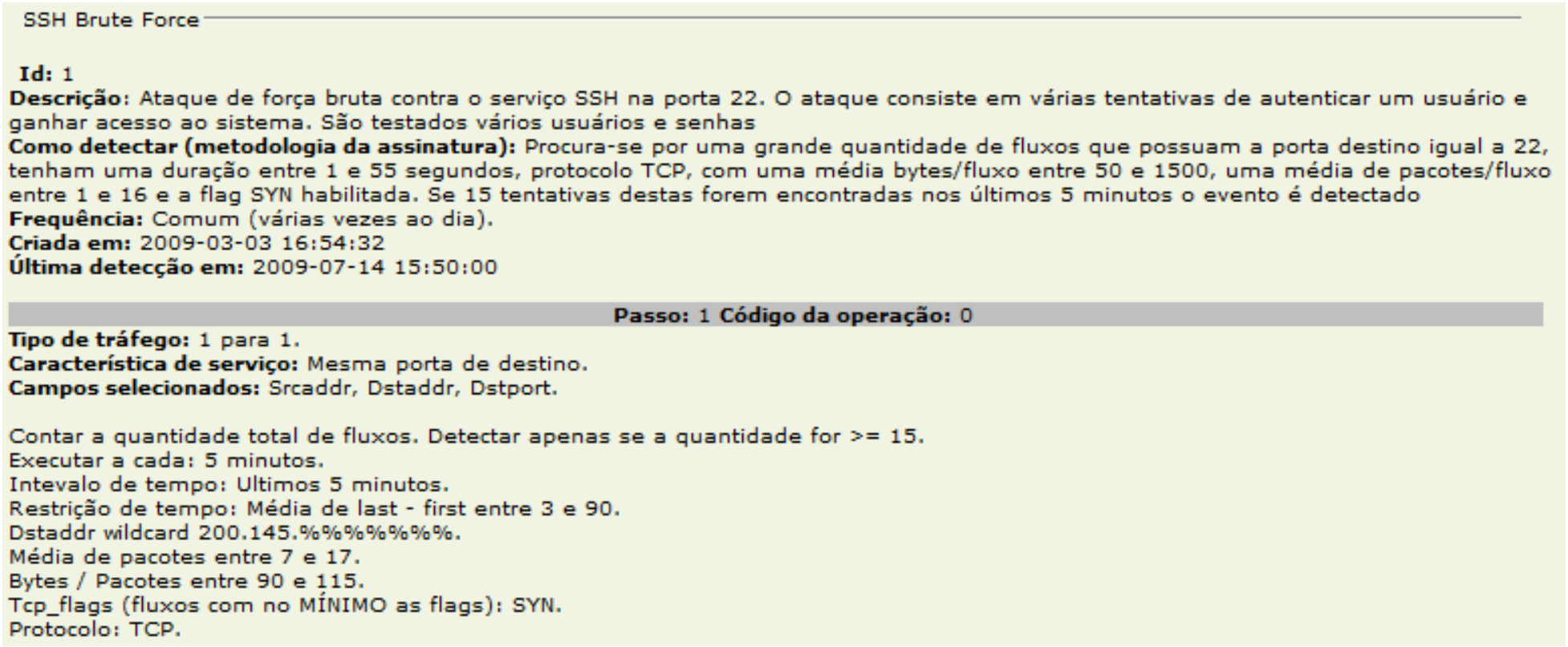

Figura 23. Assinatura para detecção de ataque de força bruta ou dicionário no serviço de SSH.

Embora na descrição da metodologia da assinatura seja mencionada a restrição da porta 22 como destino, cabe mencionar que os testes foram realizados sem esta restrição. Desta forma, fica evidente a característica do modelo de detectar eventos pelo comportamento causado nos fluxos do ambiente. Não é necessário que o serviço seja executado na porta 22, padrão do SSH. Qualquer que seja a porta utilizada, se o ataque de dicionário ocorrer ele será detectado pelas características dos fluxos gerados. Esta capacidade tem sido alvo de diversas pesquisas dentro da classificação de tráfego, tal como o modelo discutido em (Karagiannis; Papagiannaki; Faloutsos, 2005).

4.2.5 Skype

Esta assinatura detecta usuários utilizando o *softphone* Skype quando o usuário conecta e realiza alguma chamada de voz. Testes foram realizados utilizando tanto o Skype para sistemas Linux quanto para sistemas Windows, sendo ambos detectados. A assinatura é dividida em dois passos, sendo um correspondente à inicialização da aplicação (comunicação com servidores de autenticação) e outro à chamada de voz.

O primeiro passo da assinatura é a detecção da chamada de voz caracterizada pelo uso do protocolo UDP, tráfego 1 para 1, tempo de duração do fluxo maior que 30 segundos, com uma média entre 130 e 320 bytes por pacote. As portas utilizadas são normalmente maiores que 1023.

Detectado um padrão como este (possível chamada de voz em andamento), o segundo passo procura pela inicialização da aplicação, caracterizada por um tráfego 1 para N (vários servidores Skype), fluxos utilizando sempre uma mesma porta de origem e com duração menor que 60 segundos. A Figura 24 ilustra a assinatura para detecção do Skype. É possível observar pela figura que o passo 1 descreve a chamada de voz e o passo 2 o processo de inicialização e autenticação, ocorrendo uma inversão na ordem cronológica destes eventos. Isto ocorre para se otimizar o tempo de análise da assinatura. Uma vez que para cada resultado do passo 1 deve-se testar a ocorrência do passo 2, quanto menos resultados intermediários o passo 1 gerar mais rápido será o teste da assinatura, pois será menor o número de análises para o passo 2. Em casos em que os passos possibilitam esta inversão, como é o do Skype, esta pode ser realizada, otimizando assim a análise *online* dos fluxos.

Skype

Id: 4
Descrição: Detecção da aplicação Skype que tenha feito autenticação e tenha efetuado alguma chamada de voz
Como detectar (metodologia da assinatura): O primeiro passo consiste em detectar possíveis chamadas de voz em que a característica do tráfego é possuir portas altas e determinado padrão de bytes e pacotes. Para cada um destes possíveis hosts realizando uma chamada são procurados os fluxos correspondentes ao startup desta aplicação, em que ocorre uma repetição de portas origem e um padrão de bytes e pacotes
Frequência: Normal (poucas vezes ao dia).
Criada em: 2009-04-05 13:43:37
Última detecção em: 2009-05-29 21:30:00

Passo: 1 **Código da operação:** 0
Tipo de tráfego: 1 para 1.
Característica de serviço: Mesma porta de origem.
Campos selecionados: Srcaddr, Dstaddr, Srcport, Dstport.

Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 5 minutos.
Restrição de tempo: last - first >= 30.
Srcaddr wildcard 200.145.%%%%%%%%.
Srcport > 1023.
Dstport > 1023.
Bytes / Pacotes entre 130 e 320.
Protocolo: UDP.

Passo: 2 **Código da operação:** 0
Tipo de tráfego 1 para N.
Característica de serviço: Mesma porta de origem.
Campos selecionados: Srcaddr, Srcport.

Contar a quantidade de endereços destino distintos. Detectar apenas se a quantidade estiver entre 50 e 500.
Contar a quantidade de portas destino distintas. Detectar apenas se a quantidade estiver entre 50 e 500.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 30 minutos (padrao).
Restrição de tempo: last - first <= 60.
Srcaddr = srcaddr do passo 1.
Srcport = srcport do passo 1.
Protocolo: UDP.

Figura 24. Assinatura para a detecção de atividade do Skype.

4.2.6 MyDoom

O MyDoom é um artefato malicioso destinado aos sistemas Windows, capaz de se propagar automaticamente por uma rede. Uma das primeiras versões surgiu em 2004 e tornou-se popular pela rapidez na disseminação. Após a análise dos fluxos deste artefato, foi gerada uma assinatura capaz de detectar um *host* sendo infectado, passando a propagar o artefato. Existe uma diferença básica na metodologia de geração da assinatura para artefatos como este. Uma vez que se trata de softwares destrutivos, para cada amostra coletada todo o sistema operacional era reinstalado. Após a reinstalação, o sistema era infectado e os fluxos gerados pela atividade do artefato exportados e coletados.

A assinatura é composta de 4 passos: 1) fluxos que possuam exatamente 7 pacotes, com 2034 bytes cada, com a porta destino 135, protocolo TCP, as *flags* Syn, Fin, Psh e Ack habilitadas, possuindo um *host* dentro do ambiente monitorado como destino; 2) utilizando cada endereço destino do passo 1 como origem, detectar fluxos com 5 pacotes, 268 bytes, a mesma combinação de *flags* e protocolo TCP; 3) os dois passos anteriores representam uma infecção bem sucedida e o terceiro passo consiste em detectar consultas a um servidor de nomes realizadas pelo *host* infectado, ou seja, uma repetição de no mínimo 30 fluxos, com porta destino 53 e protocolo UDP; 4) o passo final consiste na busca do *host* tentando propagar o artefato, sendo detectada a repetição de fluxos com porta destino 135, *flag* Syn habilitada, protocolo TCP, em que a média de bytes por pacote seja 48 (dOctets/dPkts) em um tráfego 1 para N, característico de uma varredura. A Figura 25 mostra esta estrutura.

A complexidade na geração de assinaturas para artefatos maliciosos decai sob a dificuldade em se analisar o comportamento real destes softwares maliciosos. A geração das amostras de fluxos não é trivial como em outros eventos devido a capacidade destrutiva destes artefatos. Outro fator que aumenta a complexidade é a necessidade de se reinstalar os sistemas operacionais a cada amostra. Uma vez infectado o sistema deve ser descartado para garantir que novas amostras contenham os mesmos comportamentos, desde a infecção do *host*. Desta forma a assinatura descreverá tanto a etapa de infecção quanto a do comportamento do *host* já infectado, e não apenas esta última, como seria observado caso o sistema operacional não fosse reinstalado e uma nova amostra fosse gerada.

MyDoom

Id: 3
Descrição: Propagação do Worm MyDoom, variante A.
Como detectar (metodologia da assinatura): Consiste na detecção de 4 passos, desde a infecção dos host, transmissão de uma sequencia de bytes e pacotes, consultas de DNS e tentantiva de disseminação automática.
Frequência: Raro (não ocorre diariamente).
Criada em: 2009-07-28 22:26:30
Última detecção em: 2009-01-01 00:00:00

Passo: 1 **Código da operação:** 0

Nenhum agrupamento de tráfego.
Nenhum agrupamento de portas.
Campos selecionados: Dstaddr, Srcport, Dstport.

Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 10 minutos.
Restrição de tempo: Dstaddr wildcard 200.145.%.
Dstport = 135.
Tcp_flags (fluxos com no MÍNIMO as flags): ACK, PSH, SYN, FIN.
Protocolo: TCP.

Passo: 2 **Código da operação:** 0

Nenhum agrupamento de tráfego.
Nenhum agrupamento de portas.
Campos selecionados: Srcaddr.

Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 10 minutos.
Restrição de tempo: Srcaddr = dstaddr do passo 1.
Tcp_flags (fluxos com no MÍNIMO as flags): ACK, PSH, SYN, FIN.
Protocolo: TCP.

Passo: 3 **Código da operação:** 0

Nenhum agrupamento de tráfego.
Nenhum agrupamento de portas.
Campos selecionados: Srcaddr, Dstport.

Contar a quantidade total de fluxos. Detectar apenas se a quantidade for >= 20.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 10 minutos.
Restrição de tempo: Srcaddr = srcaddr do passo 2.
Dstport = 53.
Protocolo: UDP.

Passo: 4 **Código da operação:** 0

Tipo de tráfego 1 para N.
Nenhum agrupamento de portas.
Campos selecionados: Srcaddr, Dstport.

Contar a quantidade de endereços destino distintos. Detectar apenas se a quantidade for >= 10.
Executar a cada: 5 minutos.
Intevalo de tempo: Ultimos 10 minutos.
Restrição de tempo: Srcaddr = srcaddr do passo 3.
Dstport = 135.
Bytes / Pacotes = 48.
Tcp_flags (fluxos com no MÍNIMO as flags): SYN.
Protocolo: TCP.

Figura 25. Assinatura para detecção do *worm* MyDoom.

4.2.7 Assinatura de anomalia baseada em média acumulada

Esta assinatura refere-se à metodologia explicada na subseção 3.4.2. Trata-se de uma metodologia para a detecção de comportamentos anormais em variáveis como a quantidade de fluxos, de bytes ou de pacotes, utilizando para isso valores médios acumulados, de intervalos anteriores e o atualmente obtido da análise *online* dos fluxos do ambiente.

A Figura 26 mostra a descrição da assinatura de anomalia que utiliza este mecanismo. O parâmetro 1 indica 12 intervalos anteriores (para o cálculo da média dos intervalos) e uma porcentagem de 30% para a diferença na variável analisada, que neste caso é a quantidade de fluxos.

A Figura 27 mostra um gráfico da quantidade de fluxos do ambiente monitorado. Esta figura é gerada pelo sistema FlowScan (Dave, 2000). Pode-se ver uma janela de aproximadamente 24 horas de fluxos. Perto das 16h existe uma anomalia visível na quantidade de fluxos.

Anomalia de fluxos

Id: 1
Descrição: Detecta uma anomalia na quantidade total de fluxos do ambiente de rede, sem distinção de serviços ou entrada e saída
Como detectar (metodologia da assinatura): São mantidas as médias dos intervalos anteriores e uma média acumulada durante a operação do sistema. Se a média corrente calculada for maior que [(Média Acumulada * 2 + Média dos Intervalos)/3 + %], ou o equivalente para menor, uma anomalia é detectada
Frequência: Comum (várias vezes ao dia).
Criada em: 2009-07-02 17:10:27
Última detecção em: 2009-07-29 13:38:00

Passo: 1 Código da operação: 1

Executar a cada: 5 minutos.
Tipo de anomalia: de Fluxos.
Pesquisar intervalos de tempo de: 5 minutos.
Parâmetro 1: 12.
Parâmetro 2: 30.
Protocolo: Todos.

Figura 26. Assinatura de anomalia - metodologia de média acumulada.

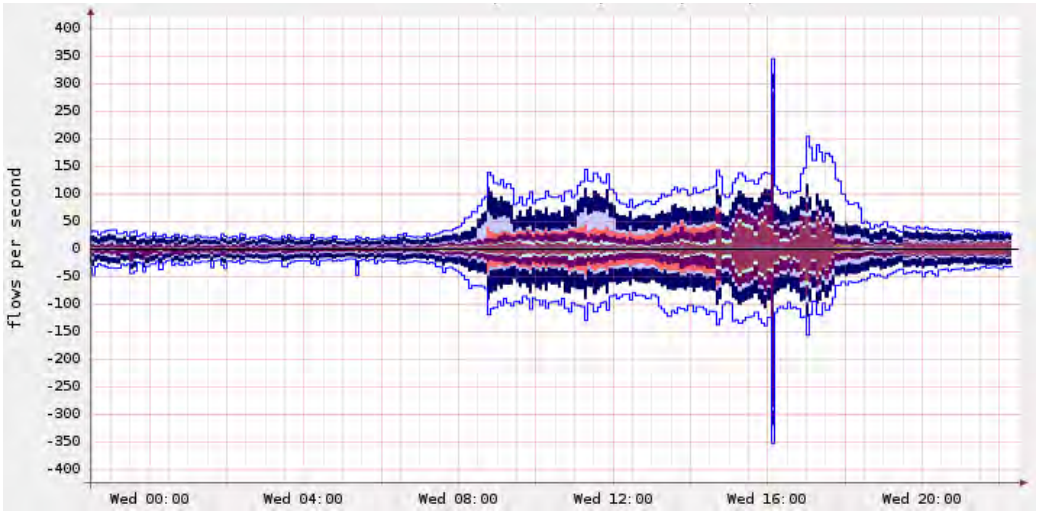

Figura 27. Visualização da quantidade de fluxos do ambiente monitorado: anomalia de fluxos visualmente detectável.

Utilizando a assinatura mencionada, o sistema foi capaz de detectar esta anomalia gerando um alerta. A Figura 28 mostra os dados deste alerta.

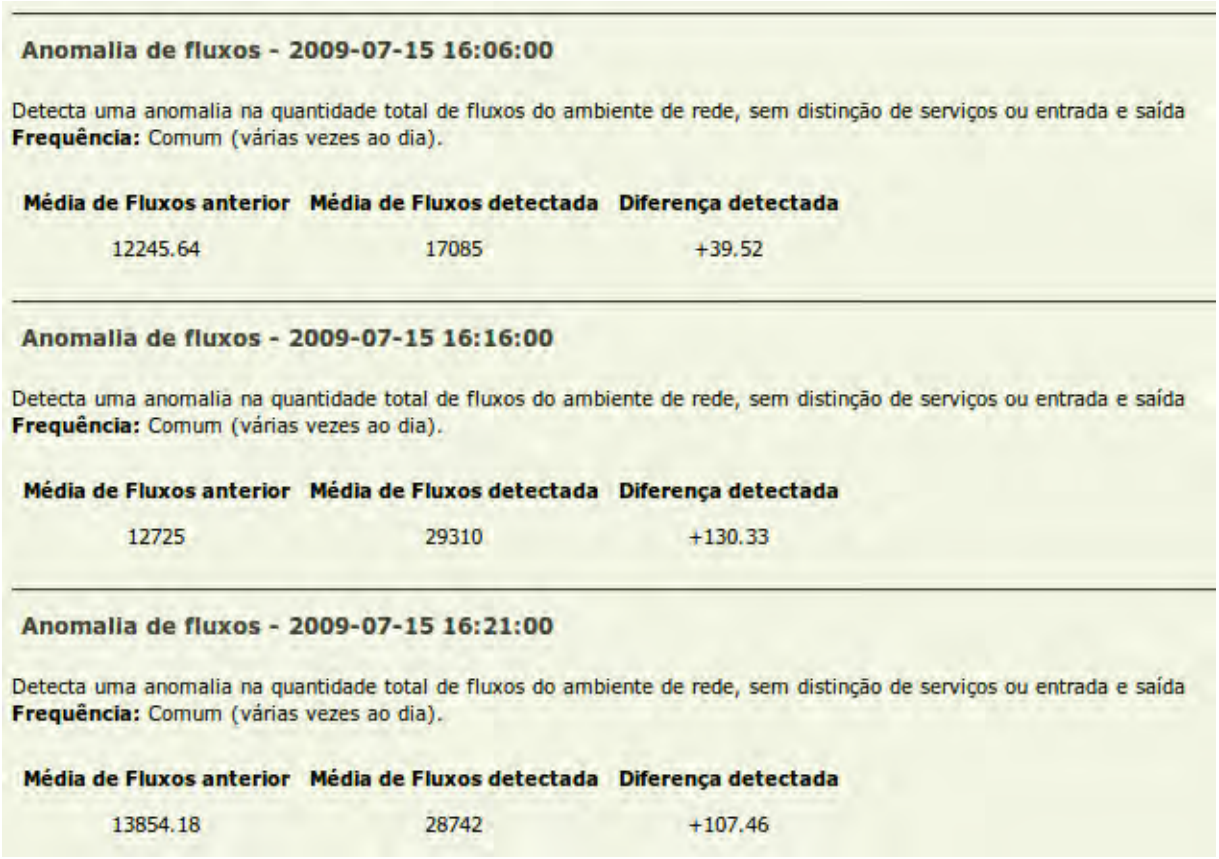

Figura 28. Detecção de uma anomalia de fluxos utilizando a metodologia de médias acumuladas.

Neste alerta pode-se verificar os dados da anomalia detectada, no mesmo horário em que aparece no gráfico. O valor *média de fluxos anterior* refere-se ao resultado da expressão que utiliza a média acumulada e a média dos intervalos anteriores. Estes valores são utilizados para que o sistema não gere falso-positivos em momentos específicos do dia, quando a quantidade de fluxos se altera devido a atividade na rede (por exemplo, as 8h e 20h do gráfico). O valor *média de fluxos detectada* refere-se ao valor da média de fluxos verificada no intervalo de tempo sendo analisado. O valor *diferença detectada* mostra, em porcentagem, a diferença verificada entre o valor momentâneo da quantidade de fluxos e o valor referência calculado da expressão mostrada na subseção 3.4.2. Neste caso, a média de fluxos aumentou, repentinamente, até 130%. Este tipo de evento pode representar diversos tipos de ataque. A importância das assinaturas de anomalia está na detecção de eventos que não são descritos nas assinaturas de abuso. Assim, o sistema possibilita a detecção de eventos imprevistos e

não descritos nas assinaturas de abuso, pela detecção de comportamentos anômalos gerados no ambiente.

Utilizando a mesma assinatura, outro tipo de evento detectado é a falha de conectividade da rede. Isto pode alertar o administrador para que as devidas ações de reparação sejam tomadas o mais rapidamente possível. A Figura 29 mostra um evento de interrupção nas transmissões da rede monitorada. É interessante notar que o evento ocorrido representa problema no envio de dados da rede monitorada para a Internet, de modo que esta continua a receber dados. Embora não se trate de uma queda geral do enlace de comunicação, a anomalia foi detectada.

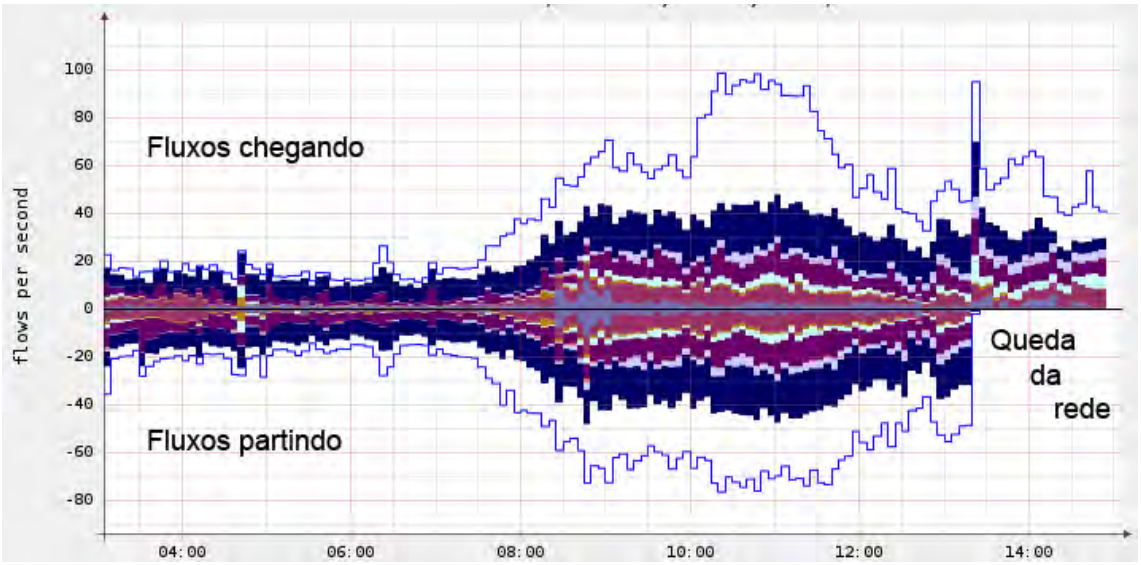

Figura 29. Visualização de uma anomalia de problema de comunicação.

4.2.8 Considerações gerais sobre efetividade das assinaturas

As assinaturas geradas detectaram os eventos para os quais foram propostas. Uma consideração quanto as assinaturas deve ser feita principalmente por impactar diretamente na efetividade do modelo de rastreamento de fluxos. As assinaturas anteriores foram geradas a partir de uma análise manual de características consideradas importantes e distintivas entre os eventos. A efetividade do modelo se relaciona diretamente com a qualidade da especificação destas assinaturas. Assim, se uma assinatura for gerada sem muito detalhamento, é provável que as taxas de erros de classificação sejam grandes. Quanto mais detalhes a assinatura contiver, maior será o grau de acerto. Portanto, a análise da efetividade do sistema deve sempre ser correlacionada com a qualidade da assinatura. O objetivo principal, desde o início deste

trabalho, era viabilizar o modelo e comprovar sua funcionalidade na análise *online* de fluxos, e não a qualidade das assinaturas geradas, apesar dos resultados bastante promissores obtidos até então.

Um exemplo que pode afetar a eficiência de uma assinatura são os sistemas de NAT (*network address translation*). Estes são sistemas de tradução de endereços que possibilitam uma rede local operar com endereços não roteáveis, comunicando-se com a Internet por meio de um ou poucos endereços IPs roteáveis, ou também conhecidos como públicos. Cada pacote da rede local tem seu endereço de origem traduzido para que seja possível o retorno dos pacotes da sessão ou conexão. Esta tradução normalmente ocorre nos limites da rede local. Desta forma, quando os pacotes de todos os *hosts* da rede local passam pelo dispositivo exportador de fluxos, como um roteador, todos os *hosts* aparecem com o mesmo IP de origem. Esta situação pode levar o modelo a cometer alguns erros, principalmente quando o evento utilizar em sua assinatura algum tipo de agrupamento, como tráfegos 1 para N ou N para 1. Quanto mais detalhada a assinatura, menor é a chance de um erro ocorrer, mesmo quando um NAT é utilizado.

4.3 Testes realizados, índices e taxas de detecção

Os resultados quantitativos foram coletados em um ambiente isolado montado propriamente para a geração destes dados. Como referência para as taxas de acertos e erros foram utilizados dois sistemas de detecção por assinaturas, que analisam conteúdo de pacotes, bastante utilizados na área e considerados de grande excelência: o Snort e o L7-Filter. O Snort, como já mencionado, é um SDI baseado em assinaturas utilizado mundialmente. O L7-Filter é uma extensão do sistema de Firewall Iptables presente nos sistemas Linux. Esta extensão permite que os pacotes sejam analisados na camada de aplicação, ou seja, seus conteúdos sejam verificados, utilizando assinaturas que descrevem uma grande quantidade de protocolos. Além da verificação, o L7-Filter permite o bloqueio *online* dos protocolos reconhecidos. A Figura 30 mostra o ambiente utilizado para a geração e coleta de informações quanto às taxas de erros e acertos.

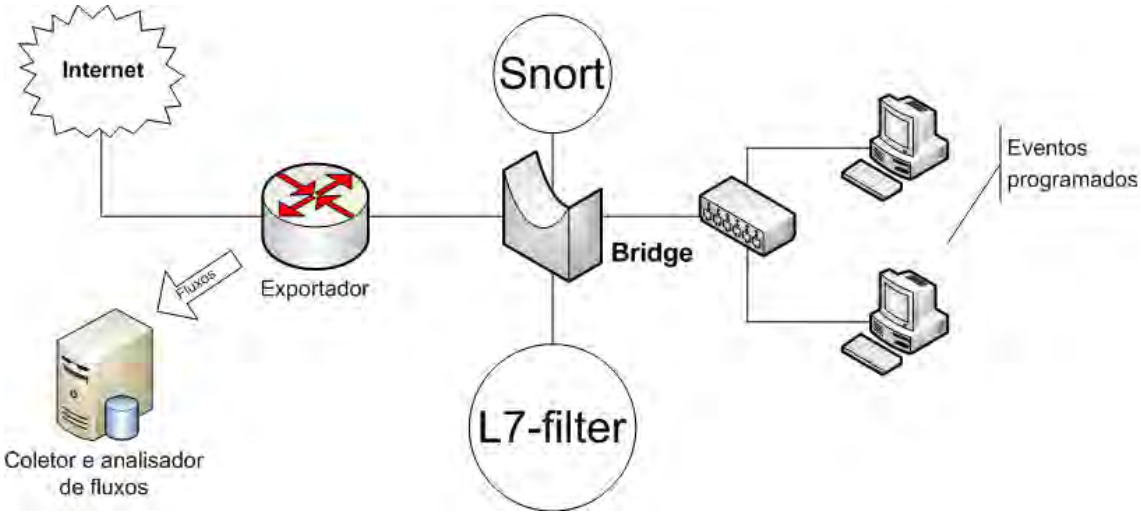

Figura 30. Ambiente para geração de eventos e testes de detecção.

Este ambiente é composto de dois *hosts* hospedeiros das aplicações a serem testadas, ou servidores de serviços a serem atacados. Uma *Bridge*, sistema que opera na camada de enlace de dados, executava os sistemas Snort e L7-Filter analisando o conteúdo de todos os pacotes originados ou destinados aos dois *hosts*. Para cada aplicação testada, quando existente, eram habilitadas as regras destes dois sistemas para que a detecção ocorresse especificamente. Por exemplo, ao testar o protocolo P2P, as regras para detectá-lo eram habilitadas. No caso de não existir regra específica para um protocolo, todas as classes de regras eram habilitadas.

Dentre todas as assinaturas geradas, foram escolhidas as mais complexas para testar a eficiência do sistema, pois estas estruturas são as que poderiam gerar as taxas mais altas de erro. Desta forma, foi analisada uma situação próxima ao pior caso, revelando a eficiência do modelo para um caso extremo. Para tanto, foram escolhidas as assinaturas de P2P e Bittorrent, para os primeiros testes. Além do fator de pior caso, estes dois eventos são facilmente gerados em um ambiente de rede. Como será observado adiante, nem todos os eventos são facilmente reproduzidos no ambiente de testes, principalmente os que necessitam de fatores externos para sua ocorrência.

Os eventos foram programados para executarem por 9 dias consecutivos, a cada 5 minutos. Este período foi utilizado por gerar uma quantidade suficientemente grande de eventos além de compreender os dias de expediente normal mais os dias de um final de semana. O teste foi iniciado em uma segunda-feira sendo executado até a terça-feira da próxima semana. Este período é importante pois permite testes em dias com comportamentos de tráfego diferentes, como ocorre nos sábado e domingos. Foi gerado

um número total de 1296 ocorrências para cada um dos eventos, durante todo o período. As respectivas assinaturas foram habilitadas nos sistemas Snort e L7-Filter. Três índices foram analisados nos resultados obtidos.

1. Taxa de acertos: é o número de ocorrências detectadas em relação ao número real de ocorrências.
2. Falso-positivo: é a ocorrência que o sistema detecta, mas que na realidade não ocorreu. Trata-se de erro cometido pelo detector quando este acusa uma ocorrência não existente.
3. Falso-negativo: é a falha quando o sistema não detecta um evento que realmente ocorreu.

O Snort não foi capaz de detectar o tráfego P2P produzido pela aplicação aMule. As taxas de acertos para cada dia de teste estão dispostas na Tabela 20. A coluna ACHoW refere-se ao modelo de rastreamento de fluxos.

Tabela 20. Taxa de acertos do Snort, L7-Filter e ACHoW para eventos P2P e Bittorrent.

TAXA DE ACERTO						
	aMule	BitTorrent	aMule	BitTorrent	aMule	BitTorrent
	Snort		L7-filter		Achow	
1º dia	-	0,92	0,99	0,95	0,92	0,89
2º dia	-	1,00	1,00	0,99	0,94	0,90
3º dia	-	1,00	1,00	0,99	0,89	0,88
4º dia	-	1,00	0,74	1,00	0,78	0,56
5º dia	-	1,00	0,83	0,99	0,81	0,69
6º dia	-	1,00	0,92	0,99	0,97	0,57
7º dia	-	0,99	0,85	0,99	0,97	0,58
8º dia	-	0,99	0,92	1,00	0,99	0,54
9º dia	-	0,88	0,94	0,90	0,81	0,35

Tanto o Snort quanto o L7-Filter possuem altas taxas de acerto, pois utilizam assinaturas baseadas em conteúdo de pacotes. Estas assinaturas normalmente descrevem a inicialização dos protocolos. Por exemplo, os primeiros pacotes P2P transmitidos são suficientes para que estes sistemas detectem a aplicação, pois as assinaturas descrevem o conteúdo, sempre com um mesmo padrão, dos pacotes iniciais.

O sistema ACHoW de rastreamento de fluxos apresentou resultados satisfatórios. Para o protocolo P2P o ACHoW apresentou taxas de acerto semelhantes as do L7-Filter e um pouco inferior ao Snort. Para o Bittorrent as taxas de acertos também foram satisfatórias. A Tabela 20 mostra uma taxa perto de 90% de acertos para os três

primeiros dias. Nos demais dias, destacados, estas taxas se mostram mais baixas. Nestes dias o protocolo operou diferenciadamente. As assinaturas de Bittorrent possuem como um dos passos necessários para a detecção o reconhecimento do tráfego referente à transmissão dos arquivos utilizando o protocolo TCP. Esta transmissão é um tráfego do tipo N para 1, em que vários *hosts* enviam partes dos arquivos para o *host* que executa o cliente Bittorrent. A diminuição das taxas de detecção nos dias destacados é devido à aplicação apenas iniciar e não baixar o arquivo ou iniciar e baixar arquivos intermitentemente. Esta característica foi verificada pois estes eventos eram controlados. Devido à natureza distribuída do compartilhamento de arquivos, nem sempre é garantido que a aplicação iniciará e começará a baixar os arquivos, sendo dependente da disponibilidade dos outros *hosts*. Assim, pode-se considerar que quando o evento ocorre normalmente ele é detectado com uma taxa de acertos satisfatória. Para os outros dois casos o sistema detecta com taxas menores de acerto.

É interessante mencionar que nestes dois casos os clientes não causam tanta interferência na rede monitorada. A detecção destes tipos de aplicação objetiva evitar problemas com direitos de cópia dos arquivos e evitar que consumam banda da rede. Quanto ao primeiro caso, as taxas menores podem indicar um tempo maior para que a aplicação seja detectada. No entanto, para o caso de consumo de banda as taxas mais baixas não implicam problema, visto que os clientes foram apenas inicializados e não estão de fato transmitindo arquivos.

A Tabela 21 mostra as taxas de falso-positivos.

Tabela 21. Taxa de falso-positivos do Snort, L7-Filter e ACHoW para eventos P2P e Bittorrent.

TAXA DE FALSO-POSITIVOS						
	aMule	BitTorrent	aMule	BitTorrent	aMule	BitTorrent
	Snort		L7-filter		Achow	
1º dia	-	0,19	0,35	0,90	0,00	0,38
2º dia	-	0,18	0,19	0,94	0,00	0,00
3º dia	-	0,24	0,36	0,91	0,00	0,15
4º dia	-	0,14	0,46	0,91	0,00	0,00
5º dia	-	0,22	0,49	0,94	0,00	0,00
6º dia	-	0,10	0,56	0,91	0,00	0,00
7º dia	-	0,22	0,44	0,90	0,00	0,00
8º dia	-	0,17	0,29	0,90	0,00	0,01
9º dia	-	0,11	0,29	0,65	0,00	0,01

A Tabela 21 mostra uma taxa zero de erros para P2P. Isto ocorre devido ao modelo de passos das assinaturas. Para que uma assinatura seja detectada, todos os

passos devem ser verificados. Para que um falso-positivo ocorra é necessário que algum tráfego gere *exatamente as mesmas características*, para *todos* os passos da assinatura. Como as assinaturas para estes dois protocolos são detalhadas, é menos provável que o sistema cometa um erro identificando um evento que não ocorreu como sendo P2P ou Bittorrent.

No primeiro e no terceiro dias é possível observar que a taxa é mais alta que nos demais. O motivo é o mesmo pelo qual as taxas de falso-positivos observadas no L7-Filter e no Snort também são altas. Os dois protocolos deste teste operam de maneira distribuída. Assim, existe a comunicação de *hosts* externos com o cliente executando o protocolo. Esta comunicação não pode ser controlada pelo cliente. Como os eventos foram agendados, cada aplicação cliente executava por 5 minutos sendo parada em seguida. Ao parar, os *hosts* distribuídos continuavam a enviar dados para o *host* cliente monitorado, no qual a aplicação não mais executava. Para os sistemas que utilizam assinaturas baseadas em conteúdo, um único pacote basta para que a detecção ocorra. Por este motivo as taxas de falso-positivos do Snort e L7-Filter mostraram-se altas, uma vez que a aplicação não estava executando, mas o tráfego continuava a ocorrer. Para o sistema ACHoW, o detalhamento dos passos das assinaturas diminuíram estas taxas, mas mesmo assim algumas ocorrências foram detectadas como no primeiro e terceiro dias.

Finalizando esta etapa de testes, a Tabela 22 mostra as taxas de falso-negativos.

Tabela 22. Taxa de falso-negativos do Snort, L7-Filter e ACHoW para eventos P2P e Bittorrent.

TAXA DE FALSO-NEGATIVOS						
	aMule	BitTorrent	aMule	BitTorrent	aMule	BitTorrent
	Snort		L7-filter		Achow	
1º dia	-	0,08	0,01	0,05	0,08	0,11
2º dia	-	0,00	0,00	0,01	0,06	0,10
3º dia	-	0,00	0,00	0,00	0,11	0,12
4º dia	-	0,00	0,26	0,26	0,22	0,44
5º dia	-	0,00	0,17	0,17	0,19	0,31
6º dia	-	0,00	0,08	0,08	0,03	0,43
7º dia	-	0,01	0,15	0,15	0,03	0,42
8º dia	-	0,01	0,08	0,08	0,01	0,46
9º dia	-	0,12	0,06	0,06	0,19	0,65

As taxas de falso-negativos são complementares as taxas de acertos. Desta forma, a partir do terceiro dia é possível observar um aumento no número de erros para o Bittorrent. Este aumento é devido aos períodos de intermitência em que a aplicação é

ligada e os arquivos não são transferidos, faltando assim um passo na assinatura. Em outras palavras, para a contagem dos eventos gerados esses períodos, mesmo sem baixar nenhum arquivo, foram considerados como ‘evento ocorrido’. Ao não detectar um dos passos o sistema ACHoW considerou o evento como não detectado, gerando o aumento da taxa de erros. No entanto, trata-se do caso em que a aplicação não gera comportamento indesejado na rede.

A Tabela 23 mostra a média das métricas analisadas para os protocolos de P2P e Bittorrent.

Tabela 23. Taxas médias das métricas analisadas para P2P e Bittorrent.

	Snort		L7-filter		Achow		Achow normalizado	
	P2P	Torrent	P2P	Torrent	P2P	Torrent	P2P	Torrent
Taxa de acertos	-	0,98	0,91	0,98	0,90	0,66	0,89	0,92
Falso-positivos	-	0,17	0,38	0,88	0,00	0,06	0,00	0,07
Falso-negativos	-	0,02	0,09	0,10	0,10	0,34	0,11	0,08

Em outra etapa de testes, representada na tabela 23 pela coluna Achow normalizado', cada evento executou por 5 minutos, sendo utilizado um espaço de 30 minutos entre duas repetições. Um intervalo de 15 minutos posterior a parada da aplicação foi considerado como o tempo necessário para que o tráfego distribuído se exaurisse, não sendo contabilizados como falso-positivos. Pode-se observar que nestes testes as taxas do ACHoW mostram-se mais satisfatórias. As taxas de falso-positivos do Snort e L7Filter também diminuíram consideravelmente para cerca de 5%.

É importante mencionar que os sistemas Snort e L7-Filter executavam na *Bridge* mostrada na Figura 30, analisando o tráfego apenas dos dois *hosts* hospedeiros. As taxas do sistema ACHoW foram geradas pela análise dos fluxos da rede inteira do Instituto, não apenas destes dois *hosts*. Embora utilizados em diferentes locais da rede, a comparação é factível. Para comparar os sistemas utilizando o mesmo ambiente, ou o ACHoW deveria analisar apenas o tráfego da *Bridge* ou o Snort e o L7-Filter deveriam analisar todo o tráfego do Instituto. Este último caso era inviável devido à necessidade de um equipamento com maior desempenho para analisar o *conteúdo* dos pacotes de toda a rede, sem interferir na latência do tráfego, além de questões políticas quanto ao acesso ao conteúdo destes pacotes. Já no primeiro caso, o sistema ACHoW analisaria os fluxos apenas do ambiente de testes, em uma quantidade que não refletiria uma análise *online* dos fluxos do Instituto. Ainda, as taxas de erros seriam nulas, pois analisando

apenas os fluxos do ambiente não haveria outros fluxos que pudessem ‘confundir’ o sistema gerando taxas de erros. Portanto, a comparação é possível pois as análises do sistema ACHoW ocorreram em um ambiente de pior caso em relação às do Snort e L7-Filter.

Outros três eventos, cujas assinaturas já foram apresentadas, foram testados no mesmo ambiente: ataque no serviço SSH, Skype e Varredura (*scan*) de *host*. Tanto o ataque de dicionário quanto a varredura de *host* não possuem assinaturas específicas nos sistemas Snort e L7-Filter. Deste modo, foram coletados apenas os dados referentes ao modelo de rastreamento de fluxos.

Esses três eventos foram agendados para executarem por 2 dias consecutivos. No total foram geradas 96 ocorrências de ataque de dicionário no SSH, 143 ocorrências de chamada de voz com Skype e 143 ocorrências de varreduras de *host*. Estes três eventos, especificamente, geram padrões de fluxos muito bem definidos. Por este motivo, as assinaturas construídas detectaram exatamente *todas as ocorrências* dos eventos reais, sem cometer erros de falso-positivos e falso-negativos.

Por fim, a classe de eventos que compreende os artefatos maliciosos, não pôde ser testada com uma ampla quantidade de ocorrências. Existe uma restrição operacional para a geração de tráfego destes eventos. As assinaturas do sistema de rastreamento de fluxos representam exatamente a ocorrência real da infecção ou propagação de um artefato. Logo, para que estes eventos sejam detectados, este deve realmente ocorrer na rede monitorada. No entanto, por possuírem uma natureza destrutiva, torna-se inviável a execução deles em um ambiente em produção. Existem sistemas capazes de controlar a disseminação de artefatos, gerando assim um ambiente de testes restrito e próprios para serem atacados, como é o caso das *honeynets*. No entanto, estes ambientes realizam determinados tipos de filtragem em alguns pacotes que interferem nos padrões de fluxos e conseqüentemente na detecção do evento. Outra restrição é que determinados artefatos atuam como *backdoors* (programas que abrem portas para conexões externas de seus administradores) abrindo brechas para que atacantes conectem e ativem determinada ação. Mesma situação ocorre para vulnerabilidades que devem ser exploradas. Estes dois últimos casos dependem de fatores externos, ou seja, da ocorrência da ativação do evento por terceiros. Assim sendo, quanto a classe de artefatos, apenas foi possível verificar que a assinatura do MyDoom construída foi capaz de reconhecer as propagações realizadas para a própria geração destas. Em outras palavras, uma vez que se trata de evento destrutivo, ao se gerar as três amostras para a análise e construção da

assinatura, os fluxos gerados foram armazenados na tabela referente ao dia da execução. Após a geração da assinatura, esta mesma tabela foi analisada, sendo detectado o evento, *junto a todos os fluxos do dia no ambiente*. Desta forma, evitou-se executar o artefato além das três vezes necessárias para geração das amostras, não obtendo assim a detecção *online* deste evento. Durante a geração das três amostras foi gerado um alerta pelo CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil - da propagação de artefato malicioso proveniente de *hosts* no qual estava sendo testado.

Este capítulo mostrou os resultados obtidos com o modelo de rastreamento de fluxos. Diversas assinaturas foram geradas com um processo cuidadoso de isolamento de tráfego e identificação de padrões em amostras coletadas de um ambiente de teste. Estas assinaturas foram testadas no ambiente de rede do IBILCE gerando taxas de acertos e erros satisfatórias.

O capítulo 5 a seguir traz as considerações finais sobre esta pesquisa, elencando as principais considerações sobre o trabalho, dificuldades e trabalhos futuros.

Capítulo 5 - Conclusões

Este capítulo traz as conclusões, considerações sobre as dificuldades encontradas durante a pesquisa e trabalhos futuros relacionados ao modelo.

5.1 Conclusões

Monitorar o andamento de uma rede é extremamente importante para a manutenção da qualidade dos serviços oferecidos e das funcionalidades para seus usuários. Para tanto, é importante detectar protocolos e eventos presentes no tráfego de um ambiente e controlá-los segundo a política de rede vigente. Este trabalho apresentou um modelo de rastreamento de fluxos, baseado em assinaturas, que permite a um administrador procurar tanto por eventos específicos quanto por anomalias em rede.

Existe atualmente um grande interesse do mercado nas pesquisas envolvendo fluxos de rede por esta tecnologia permitir que tráfegos de grande porte comecem a ser analisados. Pontos de troca de tráfego, *datacenters* e provedores de Internet são exemplos de usuário potenciais das pesquisas que desenvolvem metodologias com fluxos. Além do interesse técnico, existe também interesse político para a monitoria de tráfego em determinados canais de comunicação. A pesquisa e resultados apresentados neste trabalho elucidam dúvidas até então existentes sobre os fluxos, dentre elas a possibilidade de descrever eventos em redes e de monitorar estes eventos de maneira *online*, além de comprovar que modelos neles baseados são realmente promissores.

Um dos objetivos deste trabalho era criar um modelo que permitisse a detecção e identificação de eventos em redes por meio de fluxos. A metodologia de assinaturas baseadas em passos, capaz de representar agrupamentos de fluxos, mostrou-se bastante funcional. Ainda, testes mostraram um grau de eficiência bastante apurado na detecção dos eventos.

O desempenho do sistema no ambiente onde foi testado não apresentou insuficiência. Diversas características na implementação do sistema ACHoW foram resultados de uma busca por um melhor desempenho do modelo. No entanto,

determinados ambientes de grande porte como os pontos de troca de tráfego, *datacenters* e provedores possuem canais de comunicação com taxas de dados muito superiores à utilizada nos testes. Quando mencionado que um dos objetivos era um sistema de baixo custo, remete-se a ambientes de pequeno e médio porte. O que fica evidente neste trabalho é que o modelo de rastreamento é funcional e eficiente. Para a monitoria de redes de grande porte fica claro que os investimentos em software e hardware devem ser proporcionais à dimensão do ambiente de rede. Muito se tem falado em modelos escaláveis de análise de rede. Pelas considerações anteriores, o modelo de rastreamento em si é considerado escalável. No entanto, esta escalabilidade está mais ligada a questões da implementação do que ao modelo em si. A descrição de eventos em passos é funcional. Estender o modelo para ambientes de grande porte é possível utilizando esforços e investimentos proporcionais ao benefício que este trará à administração desses canais de comunicação. Conseguir identificar que um *host* está executando determinado protocolo em meio aos fluxos de um ponto de troca de tráfego é tarefa até então não imaginada. O modelo proposto nesta pesquisa possibilita este fim, guardadas as proporções entre o investimento e o benefício conseguido.

Ainda neste contexto, muitos autores chegaram a apontar a impossibilidade de se gerar fluxos em canais de comunicação de alta velocidade. Para que o rastreamento de fluxos funcione é necessário que estes sejam exportados para todas as sessões e conexões do ambiente. Existe uma característica dos protocolos de exportação denominada amostragem que, uma vez utilizada, não gera fluxos para todas as comunicações, mas realiza amostragem de fluxos para diminuir a carga no equipamento exportador. Esta situação foi superada por pesquisas recentes em que fluxos são gerados sem amostragem em redes Giga (Bartos *et al.*, 2008), com hardwares específicos e dedicados.

Outra característica que deve ser abordada é a adequabilidade do modelo ao novo protocolo de rede, o IPv6. A mudança da atual versão, o IPv4, para a nova versão ocorrerá dentro de pouco tempo para proporcionar a continuidade das comunicações na Internet. O modelo de rastreamento é adaptável para esta nova versão, sendo necessárias apenas alterações na arquitetura de armazenamento para que passe a suportar um novo tipo de dado, característico dos endereços IPv6 (campos *Srcaddr* e *Dstaddr* da Tabela 1, passando a ocupar 16 bytes). Toda a *metodologia* continua a mesma, sem nenhuma influência na eficiência das detecções.

Por fim, os resultados obtidos neste trabalho são considerados bastante satisfatórios e promissores para a evolução das pesquisas com fluxos de rede. Os objetivos inicialmente propostos foram alcançados. O trabalho final apresenta diversas contribuições para a área de pesquisa com fluxos de rede, sendo a maior delas, o sucesso na criação e implementação de um modelo de rastreamento de fluxos, que permite a descrição de eventos e sua detecção *online* em um ambiente de rede, usando um paradigma híbrido de classificação, com características constantes (abuso) ou com base em distúrbios causados no ambiente (anomalia).

5.2 Dificuldades encontradas

Durante o período de desenvolvimento desta pesquisa, diversas dificuldades foram encontradas. No entanto, três delas podem ser destacadas como as mais importantes e que necessitaram um tempo maior de dedicação para sua superação.

A primeira dificuldade foi na definição da arquitetura de armazenamento. Diversos testes foram realizados para determinar qual a real necessidade de velocidade o sistema deveria ter para que as classificações pudessem ocorrer de forma *online*. Inicialmente considerou-se até mesmo a hipótese de utilizar uma rede neural para fazer uma classificação inicial de fluxos, visto que estes sistemas possuem alta velocidade de classificação de padrões. No entanto, após diversas otimizações no SGBD MySQL e no coletor de fluxos, optou-se por utilizá-lo diretamente, sem prejuízo para a classificação *online* e sem o ônus dos processos de treinamentos dos sistemas neurais.

A segunda dificuldade trata do processo de geração das assinaturas. Como a análise de definição das características que identificam determinado evento é manual, o processo de geração de assinaturas torna-se bastante dispendioso. A identificação das características repetitivas nas amostras levou determinado tempo e limitou a quantidade de assinaturas geradas.

Por fim, a terceira dificuldade, ainda que relacionada com a geração das assinaturas, trata da análise dos artefatos maliciosos. Estes eventos destrutivos merecem determinado cuidado em suas execuções. Surgiu então um paradoxo quando da geração de assinaturas para estes eventos. Não se pode ligá-los diretamente em uma rede em produção. Se for usado um ambiente restritivo e controlado, pode-se alterar significativamente o modo de operação real desses artefatos sendo que esta alteração

poderia gerar uma assinatura errada que não refletiria o comportamento real do artefato. Esta restrição limitou os testes com artefatos maliciosos.

5.3 Trabalhos futuros

Estabelecido o modelo de análise de fluxos, os trabalhos futuros são possíveis em duas áreas: processador de fluxos e geração de assinaturas.

Quanto ao processador de fluxos, melhorias podem ser feitas quanto a implementação do sistema. Os SDIs normalmente denominam seus núcleos, parte de código responsável pelo reconhecimento dos eventos, de *engines*. Esses SDIs constantemente atualizam estas *engines* para obterem cada vez mais desempenho. Da mesma forma, existem algumas mudanças que se realizadas no código do processador de fluxos podem prover maior desempenho, tornando o *sistema ACHoW* mais escalável. Dentre estas mudanças estão a melhor estruturação das estruturas de dados que armazenam os resultados intermediários e a criação de um sistema de buscas capaz de otimizar as consultas realizadas. Atualmente, para cada passo de uma assinatura é realizada uma consulta. Se várias assinaturas utilizarem uma mesma consulta, esta será executada repetidas vezes. Este procedimento pode ser melhor trabalhado de forma que o processador de fluxos não repita consultas semelhantes em assinaturas e passos diferentes, diminuindo assim o processamento e o tempo necessário para a monitoria *online*.

Do ponto de vista das assinaturas, como mencionado, todas elas foram geradas segundo um processo manual de análise de fluxos para identificação de características distintivas dos eventos. A busca por estas características identificadoras de eventos pode ser realizada de maneira automática, seguindo critérios de identificação de padrões em bases de dados. Ainda, a criação de novas assinaturas pode também ocorrer *online*, de forma que uma vez detectado determinado padrão, este possa gerar uma assinatura que é imediatamente inserida no sistema, passando a fazer parte dos eventos de interesse a serem detectados.

Referências bibliográficas

(Abad *et al.*, 2004) ABAD, C. et al. Correlation Between NetFlow System and Network Views for Intrusion Detection. **SIAM International Conference on Data Mining (ICDM)**, 2004.

(Araújo; Filho, 2009) ARAÚJO, A. B.; FILHO, J. E. M. HLBR - Hogwash Light BR. 2009. Disponível em: < http://hlbr.sourceforge.net/ >. Acesso em: 27 ago. 2009.

(Astashonok, 2009) ASTASHONOK, S. NetFlow probes: fprobe and fprobe-ulog. 2009. Disponível em: < http://fprobe.sourceforge.net/ >. Acesso em: 06 ago. 2009.

(Bartos *et al.*, 2008) BARTOS, K. et al. Flow Based Network Intrusion Detection System using Hardware-Accelerated NetFlow Probes. **CESNET Conference 2008: security, middleware, and virtualization – glue of future networks**, p. 49-58, 2008.

(Bill, 2000) BILL, N. Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics. **Proceedings of the 14th USENIX conference on System administration**, p. 285-290, 2000.

(Bin *et al.*, 2008) BIN, L. et al. A NetFlow based flow analysis and monitoring system in enterprise networks. **Computer Networks,** v. 52, n. 5, p. 1074-1092, 2008. ISSN 1389-1286.

(Bonifacio *et al.*, 1998) BONIFACIO, J. M. et al. Neural networks applied in intrusion detection systems. **Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference,** v. 1, p. 205-210, 1998.

(Cansian; Corrêa, 2007) CANSIAN, A. M.; CORRÊA, J. L. Detecção de ataques de negativa de serviço por meio de fluxos de dados e sistemas inteligentes. **VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais,** v. 7, p. 125-141, 2007.

(Cédric; Mé, 2001) CÉDRIC, M.; MÉ, L. ADeLe: an Attack Description Language for Knowledge-based Intrusion Detection. **Proceedings of the 16th International Conference on Information Security (IFIP/SEC 2001),** v. 16, p. 353-365, 2001.

(Chan; Shoniregun; Akmayeva, 2008) CHAN, Y. T. F.; SHONIREGUN, C. A.; AKMAYEVA, G. A. A NetFlow based internet-worm detecting system in large network. **Digital Information Management, 2008. ICDIM 2008. Third International Conference on**, p. 581-586, 2008.

(Chapman; Zwicky, 1995) CHAPMAN, D. B.; ZWICKY, E. D. **Building Internet Firewalls**. 1 ed. O'Reilly & Associates, Inc., 1995. 517 p. ISBN 1-56592-124-0.

(Cheswick; Bellovin; Rubin, 2005) CHESWICK, W. R.; BELLOVIN, S. M.; RUBIN, A. D. **Firewalls e segurança na Internet**. 2 ed. Bookman, 2005. 186-187; 116-117 ISBN 85-363-0429-4.

(Cisco, 2009) CISCO. Introduction to Cisco IOS Netflow - A technical overview. 2009. Disponível em: < http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.pdf >. Acesso em: 22 jul. 2009.

(Claise, 2004) CLAISE, B. RFC 3954: Cisco Systems NetFlow Services Export Version 9. 2004. Disponível em: < http://www.ietf.org/rfc/rfc3954.txt >. Acesso em: 27 dez. 2007.

(Corrêa; Proto; Cansian, 2008) CORRÊA, J. L.; PROTO, A.; CANSIAN, A. M. Modelo de Armazenamento de Fluxos de Rede para Análises de Tráfego e de Segurança. **VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais,** v. 8, p. 73-86, 2008.

(Dave, 2000) DAVE, P. FlowScan: A Network Traffic Flow Reporting and Visualization Tool. **Proceedings of the 14th USENIX conference on System administration**, p. 305-318, 2000.

(Debar; Curry; Feinstein, 2007) DEBAR, H.; CURRY, D.; FEINSTEIN, B. RFC 4765: The Intrusion Detection Message Exchange Format. 2007. Disponível em: < http://www.rfc-editor.org/rfc/rfc4765.txt >.

(Dressler; Jaegers; German, 2007) DRESSLER, F.; JAEGERS, W.; GERMAN, R. Flow-based Worm Detection using Correlated Honeypot Logs. **15. GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS 2007)**, p. 181-186, 2007.

(Elmasri; Navathe, 2005) ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 4 ed. São Paulo: Pearson Education do Brasil, 2005. ISBN 85-88639-17-3.

(Escamilla, 1998) ESCAMILLA, T. **Instrusion Detection: Network Security Beyond the Firewall**. New York: John Wiley and Sons, Inc., 1998. 348 p. ISBN 0-471-29000-9.

(Fullmer, 2001) FULLMER, M. Tool set for working with NetFlow data. 2001. Disponível em: < http://www.splintered.net/sw/flow-tools/docs/flow-tools.html >. Acesso em: 15 mai. 2009.

(Geer, 2006) GEER, D. Behavior-based network security goes mainstream. **Computer,** v. 39, n. 3, p. 14-17, 2006. ISSN 0018-9162.

(Karagiannis; Papagiannaki; Faloutsos, 2005) KARAGIANNIS, T.; PAPAGIANNAKI, K.; FALOUTSOS, M. BLINC: multilevel traffic classification in the dark. **Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)**, p. 229-240, 2005. ISSN 0146-4833.

(Kruegel *et al.*, 2002) KRUEGEL, C. et al. Stateful intrusion detection for high-speed network's. **Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium**, p. 285-293, 2002. ISSN 1081-6011.

(Larsen; Galt; Richardson, 2009) LARSEN, J.; GALT, J.; RICHARDSON, W. Hogwash. 2009. Disponível em: < http://sourceforge.net/projects/hogwash/ >. Acesso em: 27 ago. 2009.

(Laurent *et al.*, 2006) LAURENT, B. et al. Traffic classification on the fly. **Computer Communication Review,** v. 36, n. 2, p. 23-26, 2006. ISSN 0146-4833.

(Meier; Bischof; Holz, 2002) MEIER, M.; BISCHOF, N.; HOLZ, T. SHEDEL-A Simple Hierarchical Event Description Language for Specifying Attack Signatures. **Proceedings of the 17th International Conference on Information Security: Visions and Perspectives (IFIP TC11)**, p. 559-572, 2002.

(Metcalfe; Julien, 2009) METCALFE, W.; JULIEN, V. Snort Inline. 2009. Disponível em: < http://snort-inline.sourceforge.net/index.html >. Acesso em: 12 Mai. 2009.

(Microsoft, 2001) MICROSOFT. Microsoft Security Bulletin MS01-033. 2001. Disponível em: < http://www.microsoft.com/technet/security/bulletin/ms01-033.mspx >. Acesso em: 30 dez. 2007.

(Microsoft, 2002) _____. Microsoft Security Bulletin MS02-039. 2002. Disponível em: < http://www.microsoft.com/technet/security/bulletin/ms02-039.mspx >. Acesso em: 30 dez. 2007.

(Microsoft, 2003) _____. Microsoft Security Bulletin MS03-026. 2003. Disponível em: < http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx >. Acesso em: 30 dez. 2007.

(Microsystems, 2009) MICROSYSTEMS, S. MySQL. 2009. Disponível em: < http://www.mysql.com/ >. Acesso em: 05 jun. 2009.

(Myung-Sup *et al.*, 2004) MYUNG-SUP, K. et al. A flow-based method for abnormal network traffic detection. **Proceedings of the Network Operations and Management Symposium (NOMS - IEEE/IFIP),** v. 1, p. 599-612, 2004. ISSN 1542-1201.

(Oberheide; Goff; Karir, 2006) OBERHEIDE, J.; GOFF, M.; KARIR, M. Flamingo: Visualizing Internet Traffic. **Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP**, p. 150-161, 2006. ISSN 1542-1201.

(Project, 2007) PROJECT, T. L. System-independent interface for user-level packet capture. 2007. Disponível em: < http://sourceforge.net/projects/libpcap >. Acesso em: 27 dez. 2007.

(Quittek *et al.*, 2004) QUITTEK, J. et al. RFC 3917: Requirements for IP Flow Information Export: IPFIX. 2004. Disponível em: < http://www.ietf.org/rfc/rfc3917.txt >. Acesso em: 27 dez. 2007.

(Registro.Br, 2009) REGISTRO.BR. Whois. 2009. Disponível em: < https://registro.br/cgi-bin/whois/ >. Acesso em: 23 jul. 2009.

(Roesch, 2009) ROESCH, M. Lightweight intrusion detection technology. 2009. Disponível em: < http://www.snort.org/ >. Acesso em: 03 jun. 2009.

(Rubin; Jha; Miller, 2005) RUBIN, S.; JHA, S.; MILLER, B. P. Language-based generation and evaluation of NIDS signatures. **Security and Privacy, 2005 IEEE Symposium**, p. 3-17, 2005. ISSN 1081-6011.

(Steven; Giovanni; Richard, 2002) STEVEN, T. E.; GIOVANNI, V.; RICHARD, A. K. STATL: an attack language for state-based intrusion detection. **Journal of Computer Security,** v. 10, n. 1-2, p. 71-103, 2002. ISSN 0926-227X.

(Stevens, 1993) STEVENS, R. The Protocols (TCP/IP Illustrated, Volume 1). In: (Ed.): Addison-Wesley Professional, 1993. cap. 1, Seção 1.3, ISBN 0201633469.

(Taylor *et al.*, 2009) TAYLOR, T. et al. FloVis: Flow Visualization System. **Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology**, p. 186-198, 2009.

(Vigna; Eckmann; Kemmerer, 2000) VIGNA, G.; ECKMANN, S.; KEMMERER, R. Attack Languages. **Proceedings of the IEEE Information Survivability Workshop**, p. 163-166, 2000.

(Wang; Wang, 2008) WANG, Z.; WANG, X. NetFlow Based Intrusion Detection System. **MultiMedia and Information Technology, 2008. MMIT '08. International Conference on**, p. 825-828, 2008.

(Xinyou; Chengzhong; Wenbin, 2004) XINYOU, Z.; CHENGZHONG, L.; WENBIN, Z. Intrusion prevention system design. **Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on**, p. 386-390, 2004.

Trabalhos produzidos durante o desenvolvimento desta pesquisa

-

Detecção de ataques de negativa de serviço por meio de fluxos de dados e sistemas inteligentes

VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Rio de Janeiro, v. 7, p. 125-141, 2007

Adriano M. Cansian, Jorge L. Corrêa

Resumo. Este artigo apresenta um novo modelo de detecção de anomalias e tentativas de intrusão baseado na utilização de fluxos de dados (padrão Netflow) e na capacidade classificatória das redes neurais. O modelo caracteriza-se pela detecção baseada no comportamento do ambiente de rede juntamente com a capacidade de absorção de conhecimento dos sistemas inteligentes. Um novo conceito de assinatura é utilizado, sendo testados diversos modelos ao longo da evolução do sistema. Ataques como DoS, DDoS e atividades de worms são rapidamente detectados, de forma automatizada e escalável para ambientes de médio e grande porte, caracterizando um efetivo modelo de monitor para redes conectadas à Internet.

-

Modelo de Armazenamento de Fluxos de Rede para Análises de Tráfego e de Segurança

VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Gramado - RS, v. 8, p. 73-86, 2008.

Jorge Luiz Corrêa, André Proto, Adriano Mauro Cansian

Resumo. O padrão IPFIX, cada vez mais utilizado por administradores de rede, permite a análise e monitoramento do tráfego de uma rede de computadores de larga escala. Suas metodologias de análise consomem baixo custo computacional se comparadas à metodologia de análise de pacotes. O IPFIX fornece uma série de especificações para a sumarização de informações da rede, mas não prevê um modelo de armazenamento dessas informações. O objetivo deste artigo é propor um modelo de

armazenamento utilizando banco de dados relacionais que seja uma base para aplicações voltadas à análise de fluxos. Essas usufruirão da versatilidade na manipulação de dados que um banco relacional oferece. Além disso, os recursos da linguagem SQL possibilitam análises de tráfego e de segurança de grande precisão.

-

Banco de dados de fluxos IPFIX para análises de tráfego e de segurança

V Conferência Internacional de Perícias em Crimes Cibernéticos (ICCYBER), 2008, Rio de Janeiro - RJ. ICoFCS - International Conference of Forensic Computer Science, 2008, ISSN 1980-1114. v. 3. p. 60-68.

André Proto, Jorge Luiz Corrêa, Adriano Mauro Cansian

Resumo. O padrão IPFIX, cada vez mais utilizado por administradores de rede, permite a análise e monitoramento do tráfego de uma rede de computadores de larga escala. Suas metodologias de análise consomem baixo custo computacional se comparadas à metodologia de análise de pacotes. O objetivo deste artigo é propor um modelo de armazenamento para o IPFIX utilizando banco de dados relacionais que possibilite uma infraestrutura para análises de tráfego e detecção de intrusão, realizadas através dos recursos oferecidos pela linguagem estruturada de consulta (SQL). Os resultados obtidos servirão como dados periciais relacionados a eventos ocorridos em redes de computadores.

-

Detectando eventos em redes utilizando um modelo de rastreamento de fluxos baseado em assinaturas

IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Campinas - SP, v. 9, 2009.

Jorge Luiz Corrêa, André Proto, Leandro Arabi Alexandre,
Adriano Mauro Cansian

Resumo. Analisando o tráfego atual de rede é perceptível uma grande variedade de protocolos gerando tráfego em diferentes densidades, tornando-se cada vez mais difícil detectar eventos específicos em meio a esta grande diversidade. Este trabalho

apresenta um modelo de detecção de eventos baseado em assinaturas que utilizam informações fornecidas exclusivamente por fluxos. Estas assinaturas são descrições exatas (de abuso) ou baseadas em limiares (de anomalias) que permitem o rastreamento de eventos em meio aos fluxos de um ambiente de rede. O sistema ACHoW é uma implementação deste modelo e possibilita a detecção e identificação de eventos como propagação de artefatos, negativas de serviços e anomalias em geral.