MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

EMANUEL JOSÉ PACHECO FREIRE

DETECÇÃO DE ANOMALIAS EM TRÁFEGO HTTP

Rio de Janeiro
2008

INSTITUTO MILITAR DE ENGENHARIA

TEN EMANUEL JOSÉ PACHECO FREIRE

DETECÇÃO DE ANOMALIAS EM TRÁFEGO HTTP

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como parte dos requisitos para obtenção do título de Mestre em Sistemas e Computação.

Orientadores: Maj Ronaldo Moreira Salles - Ph.D.
Prof. Artur Ziviani - Dr.

Rio de Janeiro
2008

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

F866d Freire, E. J. P.

Detecção de Anomalias em Tráfego HTTP/ Emanuel José Pacheco Freire. – Rio de Janeiro: Instituto Militar de Engenharia, 2008.

88 p.: il., graf., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2008.

1. Redes de computadores. 2. Medições em redes. I. Título. II. Instituto Militar de Engenharia.

CDD 004.6

INSTITUTO MILITAR DE ENGENHARIA
TEN EMANUEL JOSÉ PACHECO FREIRE
DETECÇÃO DE ANOMALIAS EM TRÁFEGO HTTP

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como parte dos requisitos para obtenção do título de Mestre em Sistemas e Computação.

Orientadores: Maj Ronaldo Moreira Salles - Ph.D.
Prof. Artur Ziviani - Dr.

Aprovada em 18 de janeiro de 2008 pela seguinte Banca Examinadora:

Maj Ronaldo Moreira Salles - Ph.D. do IME - Presidente

Prof. Artur Ziviani - Dr. do LNCC

TC Edison Ishikawa - D.Sc. do IME

Prof. Nilton Alves Júnior - D.Sc. do CBPF

Prof. Sidney Cunha de Lucena - D.Sc. da UNIRIO

Rio de Janeiro
2008

AGRADECIMENTOS

Agradeço a todas as pessoas que contribuíram com o desenvolvimento desse trabalho, tenha sido por meio de críticas, idéias ou qualquer outra forma de auxílio. Em especial, desejo agradecer aos meus orientadores, aos demais membros da banca e às pessoas citadas a seguir.

André Oliveira Castelucio

Antônio Tadeu Azevedo Gomes

Bruno de Souza Pinto Marques Correa

Emerson Magnus de Araújo Xavier

Marcos Gomes Pinto Ferreira

Marcos Vinícius de Oliveira do Couto

Marlos de Mendonça Corrêa

Reinaldo Fernandes Retto

Por fim, a todos os integrantes do Instituto Militar de Engenharia e do Departamento de Ciência e Tecnologia do Exército Brasileiro.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

DNS	*Domain Name System*
GPL	*GNU General Public License*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IANA	*Internet Assigned Numbers Authority*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
MIME	*Multipurpose Internet Mail Extensions*
MMS	*Multimedia Messaging Service*
NAT	*Network Address Translation*
P2P	*Peer-to-peer*
ROC	*Receiver Operating Characteristic*
RTSP	*Real Time Streaming Protocol*
SIP	*Session Initiation Protocol*
SNMP	*Simple Network Management Protocol*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifier*
VoIP	*Voice over IP*
WWW	*World Wide Web*

RESUMO

O presente trabalho apresenta uma metodologia para a detecção de fluxos anômalos em tráfego HTTP. Considera-se como anomalias o trânsito de algum outro protocolo pelas portas reservadas para HTTP. A metodologia foi avaliada com dados reais coletados a partir de um provedor de acesso Internet para o caso de fluxos de voz sobre IP (VoIP) presentes em tráfego HTTP. Os resultados experimentais mostraram uma boa eficiência do algoritmo para detecção VoIP, obtendo uma detecção de 90% dos fluxos possíveis de serem identificados com aproximadamente 2% de falsos positivos ou uma detecção de 100% de fluxos VoIP com uma taxa de falsos positivos de até 5%. Também foi avaliada nossa proposta em uma simulação de detecção em tempo real.

ABSTRACT

This work presents a method to detect anomalous flows in HTTP traffic. We consider an anomaly in HTTP traffic the transit of other protocols through Web ports. We validate our proposal using real-world experimental data gathered at a commercial Internet Service Provider (ISP) to detect VoIP flows using Web ports. Our experimental results show a performance of around 90% detection rate of disguised VoIP flows with a false positive rate of only 2%, whereas a 100% detection rate of VoIP flows in Web traffic is achieved with a false positive rate limited to only 5%. We also evaluate the feasibility of our proposal in a real-time detection simulation.

1 INTRODUÇÃO

Desde 1990, a Internet vem se tornando uma rede cada vez mais popular, como pode ser observado pela sua crescente presença em residências, empresas e outras instituições. Juntamente com a necessidade de cada vez mais pessoas e organizações terem acesso a essa grande rede de forma confiável e com alta disponibilidade, existem questões importantes como segurança da informação, privacidade e controle de acesso que devem ser gerenciadas. Nas organizações normalmente existe a figura do administrador de rede que é a pessoa responsável pela distribuição e controle dos recursos da rede. Disponibilizar um acesso Internet para uma rede interna pode ser considerado um problema relativamente simples, mas levar os usuários a um uso racional dos recursos da rede já é algo mais complexo.

Os dois protocolos de transporte comumente mais usados na Internet, o TCP e o UDP, utilizam o conceito de portas, que funcionam basicamente como um ponto de ligação entre o protocolo e uma determinada aplicação. Tradicionalmente, o número da porta de uma determinada conexão TCP ou UDP indica qual o protocolo de aplicação usado naquela conexão, de acordo com uma listagem gerenciada pela Autoridade para Atribuição de Números na Internet - IANA[1]. Assim, utilizando um *firewall*, o administrador da rede pode criar regras para bloquear o uso de determinadas aplicações na sua rede, identificando as aplicações pelo número das portas de destino de suas conexões TCP ou UDP.

Uma anomalia de rede pode ser definida como algo fora do padrão normal de tráfego. Mas a própria definição de normalidade neste contexto não é algo claro, o que pode ser normal em uma determinada rede pode ser considerado anormal em outra. Dependendo da situação, anomalias em redes de computadores são relacionadas a problemas como invasão, ataques de negação de serviço, incidência de vírus e programas maliciosos (*worms*, cavalos-de-tróia) ou até uma sobrecarga da rede com alguns poucos usuários transferindo grandes volumes de dados. Em (LAKHINA, 2004b), anomalias de redes são definidas

[1]IANA é um acrônimo para Internet Assigned Numbers Authority. A atribuição de cada porta está disponível no endereço http://www.iana.org/assignments/port-numbers [2007].

como mudanças significativas ou pouco comuns nos padrões de tráfego de uma rede.

Os problemas ou mudanças nos padrões de tráfego de uma rede podem ocorrer em diferentes níveis. No nível físico, eles são mais facilmente notados, principalmente quando levam a uma interrupção do serviço. Já os problemas decorrentes de um uso indevido de protocolos ou de aplicações maliciosas já não são tão aparentes. Problemas desse tipo podem levar a uma diminuição dos parâmetros de qualidade da rede, mas esses parâmetros também podem diminuir apenas com o uso normal dos recursos da rede.

Neste trabalho, é apresentada uma metodologia para detecção de anomalias em amostras de fluxos IP do protocolo HTTP e HTTPS, que correspondem aos protocolos reservados para as portas 80 e 443/TCP, respectivamente. Na verdade, o HTTPS não é um outro protocolo, mas uma combinação do HTTP com os protocolos de criptografia TLS ou SSL. Considera-se que a finalidade básica do protocolo HTTP é o acesso de documentos eletrônicos na Internet (páginas WWW), então é considerada uma anomalia o trânsito de qualquer outro protocolo pela portas reservadas para o HTTP ou o uso desse protocolo para outras finalidades. Alguns aplicativos tipo *peer-to-peer* ou VoIP já usam as portas HTTP como alternativa para transmissão de dados. No caso do uso do protocolo HTTP para outras finalidades que não o acesso de páginas WWW, pode-se citar como exemplo o uso do protocolo para transferência de grandes arquivos, da ordem de dezenas ou centenas de Megabytes. Considera-se que um fluxo IP é identificado pelos seguintes elementos: endereço IP de origem, endereço IP de destino, porta de origem, porta de destino e protocolo de aplicação utilizado.

1.1 MOTIVAÇÃO

O protocolo HTTP tem uma fração significativa do tráfego total da Internet, sendo normalmente o protocolo de aplicação mais popular (FRALEIGH, 2003). É importante para um administrador de rede conhecer os dados que estão trafegando pela sua rede, principalmente no enlace de ligação com a Internet, por questões de custo, escalabilidade e segurança, entre outros. Uma observação do conteúdo dos pacotes, além de ser algo trabalhoso, iria levantar questões de privacidade. Assim, para se manter a privacidade em relação ao conteúdo dos pacotes, é indicada uma classificação de tráfego apenas com

a observação dos cabeçalhos TCP/IP.

Uma maneira simples de classificação seria observar apenas o número da porta TCP ou UDP no tráfego gerado por determinada aplicação. Entretanto, atualmente o número das portas TCP/UDP não é mais uma fonte confiável para identificação da aplicação responsável pelo tráfego em questão (KARAGIANNIS, 2005). A popularização de dispositivos tipo *firewall* ou *proxy* fez com que muitas aplicações usassem técnicas diversas para não ter seu tráfego identificado, como a alocação dinâmica de portas ou criação de túneis em portas normalmente liberadas. Assim, o conceito de portas bem conhecidas está caindo em desuso, sendo, por exemplo, comum encontrar tráfego não HTTP trafegando pela porta 80/TCP, que por convenção está atribuída ao protocolo HTTP.

Uma ferramenta que fosse capaz de identificar um fluxo anômalo no meio de um conjunto de fluxos HTTP considerados normais possibilitaria ao administrador de rede ter um conhecimento maior sobre seu tráfego. Caso a análise seja feita posteriormente através de arquivos de captura, ela poderia ser usada para fins estatísticos. Caso a análise possa ser feita em tempo real, uma ação imediata poderia ser tomada em relação aos fluxos identificados. Como alguns exemplos de possíveis anomalias de rede que podem usar o tráfego HTTP, pode-se citar os ataques de negação de serviço, tráfego *peer-to-peer*, *streaming* e VoIP.

1.1.1 ATAQUES DE NEGAÇÃO DE SERVIÇO

Ataques de negação de serviço são causados por programas com intenções hostis que desejam impedir ou comprometer o funcionamento normal de um servidor ou elemento da rede (HANDLEY, 2006). Os ataques podem partir de uma única máquina ou de várias, no caso de um ataque distribuído. Além disso, é comum nesses ataques os pacotes terem o endereço de origem falsificado para dificultar a identificação da fonte. Existem vários tipos de ataques de negação de serviço e diversos alvos possíveis para eles, tais como roteadores e servidores WWW.

Um tipo de ataque comum que pode ocorrer em conexões TCP é o chamado *SYN flood*. Esse ataque faz um uso indevido do procedimento de conexão do protocolo TCP, ao enviar um pacote de pedido de conexão (*SYN*) sem completar o procedimento nem

cancelar o pedido. Vários pedidos desse tipo ao mesmo tempo podem fazer um servidor ficar sobrecarregado com pedidos de conexões e não responder para usuários legítimos.

Outro problema que pode ocorrer com servidores WWW são os eventos tipo *flash crowd*, que geram um grande número de requisições de usuários ao mesmo tempo para um mesmo destino. Eles também podem sobrecarregar um servidor de maneira similar a um ataque de negação de serviço, porém eles não apresentam intenções hostis e não devem ser considerados ataques.

1.1.2 TRÁFEGO PEER-TO-PEER

O tráfego P2P é composto principalmente por transferências em redes de distribuição ou compartilhamento de arquivos. Em um estudo realizado em 2003 (FRALEIGH, 2003), o tráfego P2P se apresenta como um possível tipo dominante na Internet, juntamente com o WWW. Porém, as medições de tráfego P2P são problemáticas, e sua aparente diminuição na proporção do tráfego pode apenas indicar que as aplicações se desenvolveram e seus dados agora trafegam ocultos como se fossem de outros protocolos (KARAGIANNIS, 2004a).

1.1.3 TRÁFEGO STREAMING

Normalmente, o protocolo UDP é preferido na transferência de dados tipo *streaming*. Desta forma, a aplicação tem o controle sobre a retransmissão dos dados e não fica sujeita ao controle de congestionamento do TCP, entre outros fatores. Contudo, a existência de um *firewall* restritivo no caminho pode inviabilizar o uso do protocolo UDP. Aplicativos mais recentes que usam tráfego *streaming* já procuram detectar automaticamente a presença de NAT ou um *firewall* na rede e usar a melhor opção disponível para transmissão, podendo inclusive usar HTTP. A versão 9 do Windows Media Player, por exemplo, pode usar o protocolo HTTP, RTSP ou MMS em TCP ou UDP (SRIPANIDKULCHAI, 2004).

1.1.4 APLICAÇÕES VOIP

Aplicações de voz sobre IP também podem usar portas HTTP para trafegar seus dados. Um exemplo de programa com essa característica é o Skype, um conhecido programa

capaz de detectar automaticamente as condições da rede e funcionar em vários cenários mesmo sem configuração do usuário. Ele usa um protocolo proprietário e não aberto, que tem como primeira opção o uso do protocolo UDP para transmissão. Caso isto não seja possível, ele pode usar o TCP na porta 80 ou 443 (BASET, 2006).

1.2 PROPOSTA

O trabalho proposto nesta dissertação consiste no desenvolvimento de uma metodologia para identificação de determinadas anomalias em amostras de fluxos do protocolo HTTP. Basicamente, a metodologia procura comparar parâmetros observados com valores de referência. Resultados muito desiguais sugerem a presença de uma anomalia.

Para a determinação dos valores de referência, foi realizada uma caracterização do tráfego WWW de forma empírica a partir de informações extraídas de pacotes HTTP capturados. Nessa etapa, foi definido um modelo de tráfego HTTP, baseado em modelos desenvolvidos anteriormente, com a relação dos parâmetros a serem observados.

Na comparação das distribuições empíricas com os parâmetros observados, foram usadas métricas retiradas de dois testes estatísticos: o valor χ^2 do teste Chi-quadrado (COCHRAN, 1952) e a distância D do teste Kolmogorov-Smirnov (MASSEY, 1951). As duas métricas escolhidas são aplicadas a parâmetros do modelo HTTP desenvolvido e seus resultados são comparados com os valores de referência.

Na determinação das distribuições empíricas, foram capturados dados reais de um provedor de acesso Internet e uma instituição acadêmica e foram desenvolvidos diversos programas para auxiliar na captura e tratamento dos dados. Por fim, a metodologia foi avaliada para a detecção de fluxos VoIP em tráfego supostamente HTTP. A qualidade do reconhecimento foi avaliada com o uso de curvas ROC[2], uma representação gráfica da sensibilidade versus a taxa de falsos positivos de um classificador binário, onde os diversos pontos da curva são formados variando-se o limiar de detecção. A sensibilidade consiste no número de eventos positivos corretamente identificados como tal sobre o número total de eventos positivos possíveis de serem identificados, ou em outras palavras, o valor má-

[2]ROC é um acrônimo para Receiver Operating Characteristic, um termo utilizado na área de detecção de sinais

ximo (1 ou 100%) menos a taxa de falsos negativos. A taxa de falsos positivos consiste no número de eventos negativos incorretamente identificados como tal sobre o número total de eventos negativos, ou em outras palavras, o valor máximo (1 ou 100%) menos a especificidade do classificador.

1.3 CONTRIBUIÇÕES DO TRABALHO

O trabalho pode ser visto como um primeiro passo na busca de um sistema de detecção mais genérico. A metodologia desenvolvida é original e não procura apenas tipos conhecidos de anomalias, ela indica tudo aquilo que se distanciar do seu padrão de normalidade. Embora a metodologia tenha sido avaliada apenas para o caso de fluxos VoIP em HTTP, ela não usou características específicas desse tipo de aplicação e pode ser aplicada em outros casos. Além da metodologia de detecção, podemos citar como outras contribuições do trabalho os itens descritos a seguir:

- Desenvolvimento do procedimento de captura de pacotes com o uso de algumas ferramentas já disponíveis para coleta e tratamento dos dados e o desenvolvimento de novas ferramentas para tarefas auxiliares. Os detalhes desses procedimentos são apresentados no Capítulo 3.

- Desenvolvimento de um modelo de tráfego HTTP e caracterização desse modelo a partir de diversas capturas de dados reais realizadas em um provedor de acesso Internet. O detalhamento desses resultados é apresentado na Seção 4.1.

- Resultados experimentais relativos a detecção de tráfego VoIP em HTTP que são usados para corroborar a metodologia apresentada. Os resultados experimentais, apresentados nas Seções 4.3 e 4.4, também sugerem que é possível o uso da metodologia para uma detecção em tempo real.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

O restante do trabalho está dividido em mais quatro capítulos. No Capítulo 2, é apresentada uma visão geral de trabalhos relacionados, conceitos sobre o protocolo HTTP

importantes para este trabalho e alguns modelos de tráfego HTTP desenvolvidos em trabalhos anteriores. A metodologia desenvolvida para detecção está apresentada no Capítulo 3, juntamente com o modelo HTTP e as métricas que serão usadas para detecção. No Capítulo 4 é apresentada a avaliação da metodologia. Primeiramente temos as distribuições empíricas geradas, em seguida uma avaliação preliminar das métricas e então são apresentados os resultados de detecção. As considerações finais e algumas sugestões para trabalhos futuros estão no Capítulo 5.

2 REVISÃO DA LITERATURA

Estamos interessados em desenvolver um método para identificar anomalias em tráfego HTTP, ou mais especificamente, anomalias de protocolo em tráfego HTTP. Primeiramente, vamos apresentar alguns trabalhos e as dificuldades encontradas para uma modelagem de tráfego de rede de maneira geral. Em seguida, vamos especificar alguns tipos de anomalias de rede e alguns métodos de identificação desenvolvidos anteriormente. Após uma visão geral sobre modelagem de tráfego e detecção de anomalias de rede, vamos apresentar nas Seções 2.4 e 2.5 alguns detalhes do protocolo HTTP e alguns modelos de tráfego desenvolvidos especificamente para esse protocolo. Por fim, apresentamos alguns detalhes do programa Skype, que foi utilizado para uma avaliação da metodologia.

2.1 MODELAGEM DO TRÁFEGO DA REDE

Sendo a Internet uma grande rede em constante mudança e evolução, uma caracterização da rede como um todo ou uma simulação de seu comportamento torna-se um problema desafiador (PAXSON, 1997). Além disso, seu crescimento não foi programado ou ordenado, muitos protocolos possuem falhas ou funcionalidades que podem ser exploradas por pessoas com intenções hostis, oportunistas ou maliciosas.

Uma primeira pergunta que pode ser feita por quem deseja uma metodologia de detecção de anomalias é como se pode modelar o tráfego de uma rede, principalmente uma rede complexa como a Internet. Em alguns casos, a chegada de pacotes em uma rede de dados pode ser normalmente modelada como um processo de Poisson, o que simplifica a análise (PAXSON, 1995).

O trabalho (PAXSON, 1995) apresenta alguns casos onde o intervalo entre chegadas de pacotes não é exponencialmente distribuído. Assim, processos com auto-similaridade seriam uma modelagem mais adequada que Poisson, tanto em redes locais quanto em redes mais amplas (WILLINGER, 1998). Uma particularidade da auto-similaridade é a apresentação de características semelhantes, mesmo quando observado em diferentes escalas,

de uma forma semelhante a um fractal. Neste caso, o tráfego apresentaria características semalhantes mesmo observado em diferentes escalas de tempo.

Em (LELAND, 1994), é apresentado um estudo do tráfego Ethernet, concluindo-se que ele tem características de auto-similaridade. No trabalho (CROVELLA, 1997) foi estudado especificamente o tráfego WWW, mostrando que ele também possui características de auto-similaridade e procurando determinar as causas desse comportamento.

Outros trabalhos (KARAGIANNIS, 2004b) já indicam que a modelagem com Poisson continua válida, mas em escalas de tempo da ordem de micro-segundos. Com escalas de tempo na ordem de segundos, observam-se as dependências de longo alcance e a modelagem com Poisson não seria mais válida. Mesmo com dependências de longo alcance, é importante observar que existem ganhos na multiplexação de tráfego (CAO, 2002), e o número de conexões ativas em um enlace vai alterar as características do tráfego.

Em (CROVELLA, 1997), procura-se modelar um processo auto-similar como a superposição de vários processos de renovação que possuem apenas dois estados possíveis: ON ou OFF. Desta forma, para o tráfego resultante ser auto-similar, é necessário que a distribuição dos tempos ON ou dos tempos OFF seja do tipo "cauda-pesada". No caso do protocolo HTTP, podemos dizer que, durante os períodos de atividade, o cliente pode estar transmitindo uma requisição ou recebendo uma resposta.

O ON representaria os períodos de atividade de um cliente da rede em um nível de aplicação, onde cada transmissão possui o conjunto de pacotes que formam o arquivo transmitido. O OFF seria o período de inatividade, que pode ser dividido em dois tipos principais no protocolo HTTP. Podem existir breves pausas entre o carregamento dos diversos objetos de uma página WWW, por exemplo, quando o navegador cliente acabou de receber um arquivo HTML, e está processando o arquivo para requisitar as figuras da página. Esse tipo de pausa é chamado OFF "ativo". O outro tipo seria uma pausa entre o carregamento de páginas WWW, quando o usuário analisa o conteúdo da página e faz uma nova requisição, carregando outro endereço ou seguindo um *hyperlink*, por exemplo. Esse tempo de inatividade é chamado OFF "inativo".

Em (MAH, 1997), procura-se desenvolver um modelo de tráfego HTTP baseado em arquivos de pacotes HTTP capturados. Parâmetros como tamanho da requisição HTTP,

tamanho da resposta, intervalo entre duas requisições ou número de arquivos por página WWW são modelados empiricamente para uso em ferramentas como simulações.

No trabalho (ESTÉVEZ-TAPIADOR, 2004) são estudados mecanismos para detecção de intrusão baseados na procura de anomalias no tráfego HTTP. Primeiramente são estudadas duas medidas para detecção de anomalias no nível de aplicação: o tamanho do conteúdo do pacote e o histograma desse conteúdo, ou seja, a distribuição probabilística dos caracteres presentes. De acordo com dados empíricos, a função de distribuição de probabilidade do tamanho do conteúdo do pacote pode ser usada para reconhecer alguns tipos de ataques, em conjunto com outras métricas, embora resultados melhores tenham sido obtidos com base no histograma do conteúdo do pacote. Para se determinar que um novo conjunto de dados possui ou não as mesmas características da função de distribuição de probabilidade empírica, foi utilizado o teste Kolmogorov-Smirnov (MASSEY, 1951). Em uma segunda parte, foi proposto um modelo com base em cadeias de Markov para modelar o tráfego HTTP obtendo melhores resultados.

2.2 ANOMALIAS DE REDE

Uma anomalia de rede pode ser considerada como qualquer coisa que altere o funcionamento normal da rede, ainda que seja causada por um acaso ou uso legítimo da rede. Desta forma, mudanças de configuração, pane de equipamentos, eventos tipo *flash crowd*, ataques de negação de serviço ou varredura de portas são exemplos de possíveis anomalias.

O termo *flash crowd* foi originário de um pequeno conto de ficção científica (NIVEN, 1973) e basicamente se refere a um evento de alto volume de requisições para um servidor, como se fosse um ataque de negação de serviço. A diferença principal é que o *flash crowd* não tem intenções hostis, ele é causado por inúmeros usuários legítimos que resolveram acessar o mesmo recurso no mesmo intervalo de tempo por um motivo qualquer. Mesmo assim, um determinado servidor Internet pode ser tão prejudicado quanto em um ataque. No artigo (JUNG, 2002) é apresentada uma caracterização mais precisa dos eventos *flash crowd* e alguns métodos de distinção entre eles e ataques de negação de serviço, relacionados com servidores HTTP.

O artigo (BARFORD, 2001) foi um dos primeiros a caracterizar os tipos de anomalias de volume, ou anomalias de rede. Foram definidos 3 tipos básicos de anomalias: anomalias de rede, eventos relacionados com problemas da rede, pane, falha de configuração ou roteamento; abusos como ataques de negação de serviço; e eventos tipo *flash crowd*. A captura do tráfego foi feita no nível de fluxos IP com o programa `flowscan` (PLONKA, 2000). A partir da informação gerada pelo `flowscan` o administrador da rede poderia identificar alguns tipos de anomalias. Por exemplo, um ataque de negação de serviço poderia ser identificado por um grande aumento no número médio de fluxos IP por segundo, enquanto uma falha de algum equipamento seria identificada por uma mudança brusca na taxa de bits transmitidos ou recebidos.

Em (LAKHINA, 2004a) temos uma caracterização mais extensa dos tipos de anomalias. As anomalias causadas pela rede podem ser do tipo *outage*, quando um enlace fica inoperante ou *ingress-shift*, quando o tráfego muda seu caminho, por exemplo. As anomalias relacionadas com ataques podem ser do tipo ataques de negação de serviço (de única fonte ou distribuídos), varreduras de portas (*port scan*) ou decorrentes de vírus e programas maliciosos (*worms*). As causadas por usuários podem ser do tipo *flash crowd*, fluxos alfa ou ponto multi-ponto. Um fluxo alfa consiste na transferência de um grande volume de tráfego entre dois pontos enquanto um fluxo multi-ponto tem vários destinos.

2.2.1 ANOMALIAS DE PROTOCOLO

Uma anomalia também pode ser vista como o uso de uma porta TCP ou UDP para um serviço diferente do padronizado para aquela porta. Este tipo de anomalia de rede é normalmente chamado anomalia de protocolo. Atualmente, o número das portas TCP/UDP não é mais uma fonte confiável para identificação do tráfego, devido principalmente a popularização de dispositivos tipo *firewall* ou *proxy* nas redes. A alocação dinâmica de portas e a criação de túneis em portas liberadas para serviços como HTTP tornou-se comum, e o conceito de portas bem conhecidas está caindo em desuso.

No trabalho (MOORE, 2005) é buscada uma classificação precisa da aplicação que está usando cada porta. De maneira geral, são observados o *payload* dos pacotes em busca de assinaturas pré-determinadas para os diversos tipos de protocolos, semelhante a um

sistema de detecção de intrusão. Um ponto de possível discussão é o fato da observação do *payload* dos pacotes, mesmo por um sistema automatizado, ser ou não uma invasão de privacidade. Mesmo ignorando-se esse fato, o método apresentado é trabalhoso, exige intervenção manual e um prévio conhecimento do funcionamento dos protocolos.

No artigo (BERNAILLE, 2006) é buscada uma classificação baseada apenas na informação do tamanho dos primeiros pacotes de dados de uma conexão. É necessário um conjunto de dados para treinamento e cada fluxo seria representado por um ponto em um espaço multidimensional. Após isso, a detecção poderia ser rápida e necessitaria apenas do cabeçalho dos pacotes, mas aumenta-se o risco de falsas identificações. De uma forma similar, em (MA, 2006) é feita uma comparação de três alternativas para modelagem de protocolos, uma que considera o protocolo com uma distribuição de produtos, outra baseada em processos de Markov e uma terceira que usa grafos de *substrings* comuns para a modelagem.

2.2.2 ANOMALIAS EM SISTEMAS DE DETECÇÃO DE INTRUSÃO

Quando se fala sobre sistemas de detecção de intrusão, é comum chamar de anomalia alguma ameaça, ataque ou evento que tenha intenções hostis. Nesta área, costuma-se dividir as ferramentas disponíveis para detecção em dois grandes grupos: as ferramentas que são baseadas na procura de assinaturas conhecidas de eventos hostis e as ferramentas que procuram reconhecer anomalias de maneira geral. Os métodos baseados em assinaturas procuram padrões de uso ou características marcantes nos dados analisados, eles normalmente têm boa eficiência para os ataques já conhecidos, mas não reconhecem novas ameaças com exatidão. Programas como `snort`[3] e `bro` (PAXSON, 1999) são conhecidos exemplos desse tipo de sistema.

Os métodos que buscam reconhecer qualquer tipo de anomalia procuram criar um modelo do que seria normal e observar os desvios dos dados de entrada em relação ao modelo. Como os sistemas não são perfeitos, vão existir os casos de falsos positivos ou erros tipo I (um alerta gerado para uma situação normal) e falsos negativos ou erros tipo II (nenhum alerta gerado para uma situação anormal).

[3]Disponível em http://www.snort.org/. [2007]

Um bom sistema de detecção deve procurar manter o menor valor possível para falsos positivos e falsos negativos. Como esses sistemas em geral não possuem informações específicas sobre as anomalias buscadas, aceitam-se com eles taxas de erros maiores que as obtidas usando-se sistemas baseados em assinaturas. Porém, a própria determinação da taxa de erros já se torna um problema, pois se existisse uma metodologia automatizada e confiável para identificação dos erros, ela mesma poderia ser usada nos algoritmos de detecção de anomalias para evitar os erros.

Em um programa de detecção de intrusão que procure detectar anomalias na rede, podem ser usados parâmetros como o número de conexões em um determinado intervalo de tempo, número de pacotes ou intervalo entre chegadas para um mesmo destino para o reconhecimento de atividades suspeitas. Por exemplo, em (KRUGEL, 2002) é proposto um sistema de detecção de intrusão que procura, além dos parâmetros da rede, usar conhecimento do protocolo (HTTP ou DNS) no seu reconhecimento. Em (YE, 2001b) são apresentadas diversas técnicas probabilísticas para detecção de intrusão, entre elas o teste Chi-quadrado (χ^2). No trabalho (YE, 2001a), desenvolve-se um método para detecção de intrusão baseado no teste χ^2.

Normalmente, a grande questão sobre os sistemas de detecção de intrusão é saber se vale a pena investir em um sistema que pode reconhecer novas ameaças mas apresentar vários alertas falsos (erros tipo I e II) ou se é melhor ter um sistema que reconheça ataques com precisão mesmo sem ser capaz de reconhecer novas ameaças.

2.3 MÉTODOS DE DETECÇÃO DE ANOMALIAS DE REDES

No trabalho (LAKHINA, 2004b) é estudado um método genérico para detecção de anomalias de rede. Os autores usam o termo diagnóstico de anomalias, que consiste em três passos: detecção, identificação e quantificação. A detecção é responsável apenas por reconhecer que uma anomalia ocorreu ou está ocorrendo, a identificação determina seu tipo de acordo com um conjunto de anomalias conhecidas e o passo de quantificação estima seu tamanho e o impacto na rede.

Em (BARFORD, 2002) temos um desenvolvimento adicional de (BARFORD, 2001), visando um processo de identificação de anomalias automático. Foram usados filtros

wavelets para remover do sinal capturado a variação normal ou previsível. A origem do tráfego foi o roteador de saída do campus da universidade, foram coletados dados via SNMP com intervalo de 5 minutos e foram coletados fluxos IP na taxa de 1 pacote a cada 96. A análise ou decomposição por *wavelets* utilizada no artigo organiza os dados coletados em três partes: baixa, média e alta freqüências. Um algoritmo usa a parte baixa e média do sinal normalizadas e calcula uma pontuação de desvio dentro de uma janela de tempo definida (uma janela de tempo menor é mais sensível para anomalias de pequena duração). A pontuação calculada é comparada com um limiar de detecção. Uma pequena avaliação da técnica mostrou resultados experimentais satisfatórios.

De uma maneira similar, em (ALARCON-AQUINO, 2001) é estudada a viabilidade de se usar uma transformada discreta de *wavelets* para identificar diferenças sutis ou bruscas na variância ou freqüência do sinal, com o objetivo de reconhecer anomalias.

Em (LAKHINA, 2004a) é apresentado o método de sub-espaço e análise de componentes principais (PCA) para detecção de anomalias em fluxos IP agregados. Em (LAKHINA, 2004b) o método sub-espaço e a análise de componentes principais também foram empregados, mas diretamente com dados dos enlaces e não com fluxos IP. De modo geral, as medições podem ser feitas no nível de pacote (tais como a quantidade de conexões e bytes em cada enlace) ou no nível de fluxos. Uma das vantagens do uso de fluxos é que as anomalias normalmente se tornam bem mais evidentes ao se observar um fluxo individualmente do que o enlace como um todo. As medições são simplificadas quando não se trabalha com fluxos, pois os dados podem ser coletados diretamente dos enlaces via SNMP, por exemplo.

A análise de componentes principais consiste em uma transformação linear para um novo sistema de coordenadas, classificando de acordo com a variância, com o objetivo de facilitar a separação do considerado normal e anômalo. As primeiras n componentes formariam o sub-espaço normal e as componentes restantes o sub-espaço anômalo. Uma vez feita essa separação, o problema de identificação e quantificação são tratados de um ponto de vista vetorial.

Como seqüência do trabalho desenvolvido, o artigo (LAKHINA, 2005) apresenta uma extensão do método sub-espaço com uma análise em múltiplas dimensões e com o uso de

entropia como ferramenta adicional. A entropia já havia sido proposta como ferramenta para detecção de anomalias em trabalhos como (LEE, 2001), mas em diferentes contextos. As características do tráfego observadas são os endereços IP e as portas origem e destino.

Uma anomalia seria algo responsável por uma mudança na distribuição de valores de alguma dessas características, alterando a entropia do conjunto de dados. Para o diagnóstico, é feita uma extensão do método sub-espaço de (LAKHINA, 2004b), trabalhando com várias dimensões, uma para cada característica do tráfego observada. Para a etapa de classificação, foram usados algoritmos que agrupam os dados em *clusters*, procurando uma metodologia automatizada que não fique limitada aos tipos conhecidos de anomalias. Os resultados obtidos mostram que um certo número de anomalias, mesmo sendo de pequeno impacto, conseguem ser reconhecidas com o uso da entropia. Existem, porém, anomalias que são detectadas apenas pelas técnicas de volume, ou seja, os dois métodos apresentados são necessários para uma maximização dos reconhecimentos.

O trabalho (XU, 2005) procura desenvolver uma metodologia para reconhecer e classificar perfis de tráfego na Internet, realizando uma classificação automatizada que pode ser usada para a detecção de anomalias. Assim como em (LAKHINA, 2005), a entropia também é usada, mas com um objetivo diferente. Os fluxos IP são agrupados em *clusters* e são extraídos os *clusters* significativos usando-se entropia. Os valores que forem aparentemente aleatórios, não são levados em consideração. Com o uso de técnicas da teoria da informação, os *clusters* são classificados em classes de comportamento. As anomalias seriam então indicadas por comportamentos raros ou desvios de comportamento.

De maneira geral, quando se captura o tráfego da rede obtém-se um grande volume de informação muitas vezes irrelevante. Então, a metodologia comumente empregada para detecção de anomalias consiste em uma redução de dimensões e posterior análise do conjunto de dados. O artigo (LI, 2006) apresenta o uso de *sketches* (esboços) como técnica para redução de dimensões dos dados estudados. A agregação em fluxos origem-destino e a análise de componentes principais (PCA) usados em outros trabalhos têm objetivos similares. Neste caso, os *sketches* são conjuntos aleatórios de fluxos IP, usados juntamente com o método de sub-espaço com o objetivo de aumentar sua eficácia. A principal suposição é que a agregação aleatória de fluxos IP não vai alterar de maneira

significativa as características do tráfego do sub-espaço normal. É feita uma comparação com (LAKHINA, 2005) procurando apresentar melhorias.

Outra linha de ação consiste em usar a matriz de tráfego da rede para detectar anomalias. A matriz de tráfego é uma representação do volume de tráfego médio, em um determinado intervalo de tempo, entre todos os possíveis destinos e origens na rede. Com um conjunto de matrizes de tráfego, torna-se relativamente simples modelar e filtrar o que seria um comportamento normal de tráfego, para analisar os valores residuais. Mas a própria estimação da matriz de tráfego já é um problema complexo.

No trabalho (SOULE, 2005), a partir dos dados coletados via SNMP, é feita uma estimativa da matriz de tráfego, de acordo com a técnica dos filtros Kalman. A matriz de tráfego é comparada com uma predição feita um intervalo de tempo (5 min) antes e as divergências são analisadas a procura de anomalias usando-se técnicas diversas. Para uma melhor comparação das técnicas propostas, foram usadas curvas ROC. Essas curvas se aplicam a um classificador binário, relacionando o número de positivos verdadeiros com falsos positivos.

Em (ZHANG, 2005) também são apresentadas algumas técnicas que usam matriz de tráfego. Como a determinação da matriz de tráfego não é uma operação exata, vão existir erros no processo. Para evitar uma possível propagação de erros para a etapa seguinte, é apresentada uma técnica onde primeiro são filtrados os dados considerados normais dos enlaces e então é determinada a matriz de tráfego. São analisados vários métodos para busca de anomalias com séries temporais, análise de Fourier, *Wavelets* e PCA. Por fim, também é apresentado um sistema de detecção que possibilita mudanças de roteamento na rede, sem considerar tais mudanças como anomalias.

2.3.1 DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO

Em ataques de negação de serviço, um primeiro passo importante é conseguir reconhecer a ocorrência de um ataque. A partir daí, pode-se buscar informações adicionais, como por exemplo, saber se o ataque é distribuído ou tem uma única origem.

O trabalho (MOORE, 2001) mostra uma visão geral sobre os ataques de negação de serviço e apresenta uma técnica simples de inferência do volume de ataques no tráfego

atual baseada em respostas de ataques para endereços IP inexistentes. As premissas são que os atacantes usam endereços IP falsos escolhidos aleatoriamente e que todos os pacotes não solicitados capturados foram gerados por ataques. Muitos ataques, porém, usam técnicas de reflexão, falsificação de endereços baseada em sub-rede ou não mascaram os endereços. Nesses casos, essa técnica não seria eficiente.

Em (HUSSAIN, 2003) temos uma classificação mais elaborada, que procura diferenciar ataques simples de ataques distribuídos, com técnicas de análise espectral, entre outras. Assim, mesmo que sejam usadas máquinas refletoras ou sejam forjados os campos do cabeçalho, um ataque poderia ser classificado automaticamente como simples ou distribuído.

2.3.2 DETECÇÃO COM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS

A idéia básica dos sistemas imunológicos artificiais ou sistemas naturais é usar conceitos da biologia para a segurança de redes. Uma das possibilidades é a utilização dos chamados anticorpos artificiais, ferramentas computacionais que procurem variações entre um modelo qualquer e os dados observados. Esses anticorpos gerados podem ser regras ou amostras de tráfego geradas a partir do tráfego normal, para uma caracterização positiva, ou gerados a partir do tráfego anômalo, caso se deseje uma caracterização negativa. Algoritmos genéticos e outras técnicas também podem ser usadas para classificação.

O trabalho (DASGUPTA, 2002) apresenta um técnica de detecção onde foram usados para a codificação dos anticorpos parâmetros gerais do tráfego como número de bytes por segundo, pacotes por segundo e pacotes ICMP por segundo. Cada conjunto é considerado como uma série de tempo e o número de valores considerados varia de acordo com o tamanho de uma janela deslizante. Em (HARMER, 2002) são usados campos do cabeçalho IP dos protocolos mais comuns (TCP, UDP, ICMP) como endereço IP, portas, tamanho, *flags* para geração de uma seqüência de bytes representando um anticorpo. Para verificar a similaridade entre um anticorpo e uma amostra de tráfego podem ser usadas medições como a distância euclidiana para vetores de dados ou a distância de Hamming para vetores binários. Uma comparação das técnicas foi apresentado em (SEREDYNSKI, 2007).

A principal vantagem dos sistemas imunológicos naturais são as características de memória e aprendizado. Nos sistemas artificiais, essa vantagem é diminuída pois o apren-

dizado normalmente não é contínuo, e sim realizado dentro de uma fase definida. Em outras palavras, existe uma fase de aprendizado (onde as anomalias não devem existir) e uma fase de testes (onde as anomalias são detectadas).

2.4 O PROTOCOLO HTTP

O protocolo HTTP tem sido usado na Internet desde 1990. Uma versão inicial do protocolo, denominada posteriormente por HTTP 0.9, na verdade pode ser vista como um subconjunto da especificação completa HTTP/1.0. A especificação 1.0 evoluiu entre 1993 e 1996 até ser publicada (BERNERS-LEE, 1996). Alguns problemas da especificação 1.0 levaram ao desenvolvimento da versão 1.1, definida formalmente em (FIELDING, 1999) e que persiste até o presente momento. Vamos apresentar em seguida algumas características do protocolo HTTP 1.0 e 1.1 de interesse para a definição de um modelo HTTP.

O protocolo é constituído por mensagens de requisição ou resposta e funciona normalmente sobre conexões TCP. No seu funcionamento normal, cada cliente HTTP, também chamado agente do usuário, estabelece uma conexão com um servidor, faz uma requisição e aguarda uma resposta. Não existe resposta sem uma requisição anterior, mas uma requisição pode não ter resposta, por exemplo, se o servidor estiver muito ocupado. Um servidor Web apenas responde a requisições HTTP sem fazer outro tipo de comunicação. Outro detalhe importante é que o servidor não precisa guardar nenhuma informação sobre a requisição após o fim da conexão, ou seja, para o servidor, as requisições podem ser vistas como eventos independentes.

As mensagens HTTP tem uma estrutura simples, mostrada na FIG. 2.1. A primeira linha de uma mensagem de requisição deve ser a requisição propriamente dita. Ela é composta por um método (ou comando), um endereço (no formato URI) e a versão do protocolo. As linhas seguintes são cabeçalhos opcionais, subdividido em três partes. A primeira parte é um cabeçalho geral com informações como a hora atual. Em seguida vem o cabeçalho da requisição (ou resposta) com informações específicas do navegador cliente (ou do programa servidor) e o cabeçalho da entidade, com informações como o tamanho e tipo de dados transmitidos. Os dados, quando presentes, seguem separados do cabeçalho

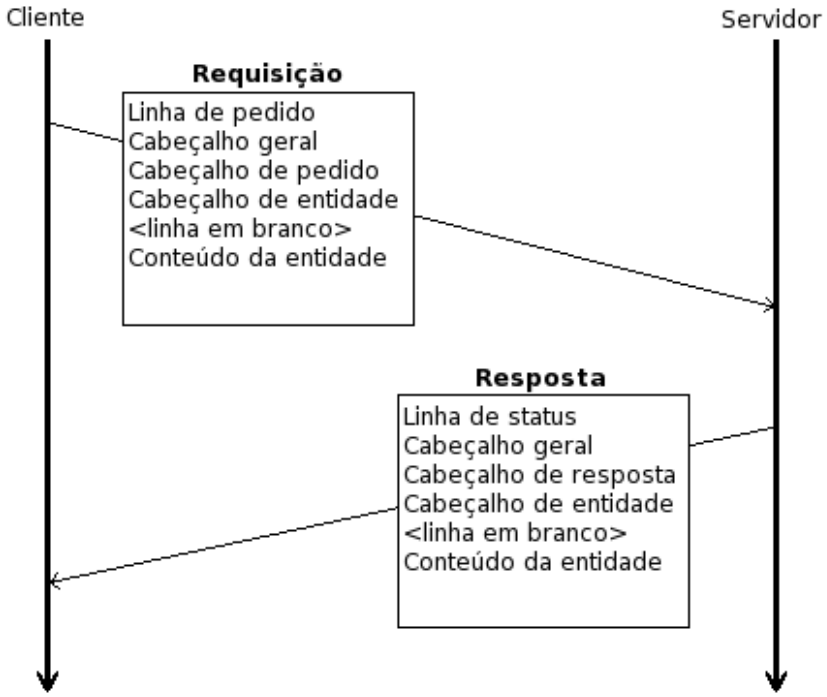

FIG. 2.1: Estrutura de mensagens HTTP.

por uma linha em branco.

As mensagens de resposta seguem um padrão semelhante. Na primeira linha tem-se a versão do protocolo, um código numérico e uma frase descrevendo se o comando foi executado com sucesso ou se houve algum erro. A listagem dos códigos, dos possíveis campos do cabeçalho e suas descrições está disponível na documentação do protocolo (BERNERS-LEE, 1996; FIELDING, 1999). O HTTP 0.9 não fazia uso de cabeçalhos, toda requisição continha apenas uma única linha com o método e o endereço. As respostas também não tinham cabeçalhos e o tamanho das respostas era indicado pelo fechamento da conexão.

2.4.1 HTTP/1.0

Em sua versão 1.0, cada conexão deveria ser aberta pelo cliente antes do envio da requisição e fechada pelo servidor após o envio da resposta. Ela também podia ser fechada antes disso, por ação do usuário, *timeout* ou outros motivos, terminando a requisição.

Os métodos mais comuns da especificação 1.0 são GET, HEAD e POST. O método HEAD é usado quando se deseja uma resposta sem conteúdo, apenas com o cabeçalho. GET é o método comum para requisição de documentos, sendo o único método do HTTP/0.9. O método POST possibilita o envio de informações para o servidor, geralmente usado em páginas com formulários.

Como já mencionado, o HTTP não trabalha com estados, ou seja, o servidor não guarda informações sobre os clientes de uma requisição para outra. No HTTP/1.0, todos os campos do cabeçalho são opcionais mas, se houver um conjunto de dados (*payload*) na requisição, o tamanho desse conteúdo deve ser informado no cabeçalho HTTP (campo *content-length*). Uma requisição pode ser feita apenas com a primeira linha, mas normalmente os navegadores sempre enviam o cabeçalho de requisição com as informações sobre o cliente. Essas informações vão ser sempre as mesmas para um mesmo cliente, mas como o servidor não deve armazenar informações entre requisições, elas são enviadas em toda requisição.

Outras funcionalidades do HTTP/1.0 incluem mecanismos para autenticação, *cache* e conexões persistentes. Na prática, um cliente necessita de muitos objetos do mesmo servidor, por exemplo no caso de uma página HTML com várias imagens. Cada imagem é carregada com uma própria requisição HTTP. Com a política de desconexão no final de cada resposta, o cliente perde algum tempo e recursos da rede com o estabelecimento de várias conexões TCP.

Para tentar melhorar o desempenho, alguns navegadores estabelecem várias conexões simultâneas com o servidor após a primeira resposta. Outras implementações começaram a usar linhas de cabeçalho para indicar que a conexão deveria ser mantida após uma resposta, mas isso nunca fez parte da especificação 1.0. O HTTP/1.1, por sua vez, já usa conexões persistentes como padrão.

O HTTP/1.0 também possui funcionalidades para *cache* local que podem evitar a retransmissão de objetos. O cliente que já carregou um determinado objeto no passado pode fazer uma requisição para ele novamente indicando a data no cabeçalho (campo *if-modified-since*). O servidor irá transmitir o objeto novamente se ele tiver sido modificado, caso não tenha sido, é enviado apenas um cabeçalho resposta com o código de resposta

304 (*not modified*).

2.4.2 HTTP/1.1

O protocolo HTTP/1.1 está especificado em (FIELDING, 1999). Uma comparação com seu predecessor pode ser vista em (KRISHNAMURPHY, 1999). Basicamente, esta versão procura melhorar as características do HTTP/1.0 com relação a servidores *proxy*, *cache*, *hosts* virtuais, entre outros. Além disso, ele trabalha com conexões persistentes (ou *keep-alive*) como padrão. Após a resposta de uma requisição a conexão (TCP) continua aberta esperando outra transação. Quando uma das partes deseja encerrar a conexão, é incluída uma linha de cabeçalho específica na mensagem.

Para facilitar o uso de *hosts* virtuais baseados em nome, o HTTP/1.1, diferentemente do 1.0, requer que o campo de cabeçalho *Host* esteja presente em todas as mensagens de requisição. Assim, uma requisição mínima HTTP/1.1 deve ser composta por duas linhas.

Para se determinar o tamanho do *payload*, além dos métodos previstos no HTTP/1.0 (uso do campo *content-length* e desconexão), pode ser usado o método *chunked*. Em alguns casos, principalmente com páginas de conteúdo dinâmico, um servidor pode não saber previamente qual o tamanho da resposta enviada, ou seja, não pode usar o parâmetro *content-length*. Como a conexão é persistente, ele também não pode desconectar após o envio. O HTTP/1.1 resolve esse problema dividindo o conteúdo da entidade (*payload* da mensagem) em vários pedaços (*chunks*) de tamanho arbitrário. Antes de cada pedaço, é enviado uma informação sobre seu tamanho, e o final da mensagem seria indicado por um tamanho zero. É usado o campo do cabeçalho *transfer-encoding* para indicar o uso dessa divisão no conteúdo da entidade.

2.5 MODELOS DE TRÁFEGO HTTP

Vamos apresentar alguns modelos de tráfego HTTP desenvolvidos para uso em simulações ou geradores de tráfego. Vamos considerar uma página ou documento HTTP como o conjunto completo, formado por um ou vários objetos carregados do servidor. Cada objeto, por sua vez, vai ser um arquivo do servidor, podendo ser uma imagem, um arquivo HTML ou de qualquer outro tipo.

2.5.1 MODELO DE ARLITT E WILLIAMSON

Um dos primeiros trabalhos de caracterização de tráfego HTTP foi (ARLITT, 1995), desenvolvido entre 1994 e 1995. Nele foram capturados apenas os cabeçalhos de pacotes TCP com as *flags* SYN ou FIN ativadas. Com o protocolo HTTP 1.0 (ou HTTP 0.9) em uso, cada objeto em uma página vai gerar uma conexao TCP para sua transferência. Na época do trabalho, foi observado que 95% das conexoes TCP transferiram menos de 42kb de dados.

O modelo usado teve 6 parâmetros:

- Número de páginas por sessão;
- Intervalo entre páginas;
- Destino da conexão;
- Número de objetos por página;
- Intervalo entre conexões;
- Bytes transmitidos em uma conexão;

Para a determinação do intervalo entre páginas, existem diferentes possibilidades: pode-se considerar o intervalo de tempo entre páginas, ou seja, entre o final de uma e o início da seguinte, ou o intervalo de tempo entre o início de uma página e o início da página seguinte, que foi a escolha dos autores. O destino da conexão foi inserido devido ao fato que, depois de uma conexão com determinado servidor, torna-se mais provável uma nova conexão com o mesmo servidor num futuro próximo.

2.5.2 MODELO DE MAH

Em 1997, Bruce Mah (MAH, 1997) desenvolveu um modelo empírico e bem detalhado de tráfego HTTP. Os dados para a formação do modelo foram obtidos com a captura de cabeçalhos TCP/IP de pacotes HTTP. Uma desvantagem apontada é que não são capturadas informações de mais alto nível como o nome dos arquivos carregados, informação essa que estaria no cabeçalho HTTP, que faz parte do *payload* TCP.

Foram retirados do arquivo de captura pacotes que exibiam uma certa periodicidade, indicando um carregamento automático de algumas páginas. Na época do artigo, o modelo HTTP/1.1 não estava totalmente padronizado e o trabalho foi baseado na especificação 1.0, com uma conexão TCP para cada objeto carregado.

O modelo usado teve os seguintes parâmetros:

- Tamanho da requisição;
- Tamanho da resposta;
- Número de objetos por página;
- Intervalo entre páginas;
- Número de páginas carregadas;
- Destino da conexão;

O intervalo entre páginas é considerado como o tempo entre o final de uma página e o início de outra. Para o número de páginas carregadas, considera-se as páginas consecutivas carregadas de um mesmo servidor, assumindo-se que todos os objetos de um documento HTTP vêm do mesmo servidor.

Foi indicado no trabalho que o tamanho da primeira requisição e da primeira resposta deveriam ser modelados cada um como um parâmetro diferente, pois eles tendem a ser maiores que as transferências seguintes. A primeira resposta normalmente é um arquivo HTML, mas também pode ser um arquivo binário ou uma mensagem de erro HTTP. As respostas seguintes normalmente são imagens pequenas.

Para se determinar o número de objetos por página, assume-se que um mesmo cliente (endereço IP) não vai carregar dois objetos não relacionados do mesmo servidor em um determinado intervalo de tempo. Assim, duas conexões que fossem separadas por um tempo maior que o intervalo definido seriam consideradas páginas distintas. Se o tempo fosse menor, seriam diferentes objetos de uma mesma página. No trabalho, o intervalo considerado foi 1 segundo.

2.5.3 MODELO DE CHOI E LIMB

Choi e Limb (CHOI, 1999) desenvolveram um modelo de tráfego HTTP usando informações tanto do cabeçalho TCP/IP como do cabeçalho HTTP. Para a definição do modelo, foi realizada uma captura dos primeiros 300 bytes do conteúdo dos pacotes TCP. De uma forma similar ao apresentado em (MAH, 1997), o primeiro par requisição/resposta teve um tratamento diferenciado. Foi formado um grupo apenas com as primeiras requisições e outro com as demais. As primeiras requisições foram consideradas responsáveis pelo carregamento do objeto principal de um página Web, ou seja, um arquivo HTML ou similar (php, asp, cgi, etc). As demais requisições são responsáveis pelos chamados objetos inseridos, ou seja, imagens e outros objetos.

As primeiras requisições são identificadas pela extensão do arquivo ou pelo tipo MIME da resposta (*text/html*) juntamente com código 200 (OK). O procedimento não é explicado em detalhes, mas são definidas regras de identificação. Assim, no caso de duas conexões quase ao mesmo tempo de um mesmo cliente para um mesmo servidor, os dois arquivos HTML devem ser considerados requisições distintas, enquanto no caso de uma página que possui *frames*, os vários arquivos HTML devem ser considerados da mesma requisição.

Em relação ao mecanismo de *cache* HTTP, se um objeto está no *cache* e seu período de validade não expirou ainda, ele é considerado válido. Se um objeto estiver no *cache* e já tiver expirado, ele deve ser validado ou não pelo servidor. Se uma resposta tiver o código 304, deve ser usado o objeto presente no *cache*. Baseado nessa definição, considera-se no modelo que a requisição como um todo foi carregada do *cache* se a maioria de seus objetos tiver sido recuperada dele.

O modelo usado tem os seguintes parâmetros:

- Tamanho da requisição;
- Tamanho do objeto principal;
- Tamanho dos objetos inseridos;
- Número de objetos por página;
- Tempo de inatividade entre requisições;

- Intervalo entre objetos inseridos;
- Tempo para processar o arquivo HTML (*parsing time*);
- Número de requisições consecutivas carregadas do *cache*;
- Número de requisições consecutivas não carregadas do *cache*;

Para a determinação do número de objetos por página, contam-se apenas os objetos inseridos e que são carregados, não levando em consideração o arquivo HTML (primeira resposta) nem os objetos carregados do *cache*.

2.6 O PROGRAMA SKYPE

O programa Skype é uma das aplicações que adotam a técnica de usar portas HTTP para enviar seus próprios fluxos como uma forma de enganar *firewalls* ou outros dispositivos de rede. O Skype é uma aplicação VoIP bastante popular que teve um grande aumento de usuários nos últimos anos. Ele usa um protocolo proprietário e fechado para comunicação entre usuários Skype, ou seja, praticamente não existe documentação disponível para o público sobre seu funcionamento. O Skype realiza chamadas de voz sem usar SIP ou algum outro conhecido protocolo de sinalização. Além disso, a comunicação entre seus vários clientes se assemelha a uma rede *peer-to-peer*, sem a necessidade de um servidor central para comunicação.

Pela experiência de uso, sabe-se que ele é capaz de funcionar em vários ambientes de rede com poucas ou nenhuma configuração do usuário. O programa consegue reconhecer automaticamente as características da rede ou usar outros computadores com o programa Skype em execução para redirecionar seu tráfego. Sabe-se também que ele pode usar as portas HTTP para envio de tráfego VoIP.

Devido a sua grande popularidade e a uma falta de informação detalhada sobre seu funcionamento, as características do programa Skype e seu protocolo foram alvo de trabalhos recentes. De maneira geral, esses trabalhos deduzem as características do programa a partir de resultados experimentais. Uma análise do comportamento do programa durante a fase de *login*, estabelecimento de uma ligação, passagem por um *firewall* e outras

operações é apresentado em (BASET, 2006). O trabalho (GUHA, 2006) consiste em estudos experimentais do tráfego Skype com resultados colhidos durante um período de 5 meses. Esses dois artigos também observaram em suas versões estudadas que o programa Skype não realizou supressão de silêncio, isto é, foi gerado um tráfego aproximadamente constante durante uma ligação VoIP estando as partes envolvidas em silêncio ou em conversação.

Em (SUH, 2006) os autores apresentam de maneira geral como reconhecer e caracterizar tráfego redirecionado apresentando resultados para o caso do Skype. É apresentada uma heurística de identificação de tráfego Skype que usa informação do conteúdo dos pacotes e uma técnica para detecção de tráfego redirecionado. Em (EHLERT, 2006) o tráfego Skype é analisado procurando-se desenvolver assinaturas de tráfego que possibilitem sua identificação. O recente trabalho (BONFIGLIO, 2007) apresenta duas abordagens para detecção de tráfego Skype em fluxos TCP ou UDP. Um dos métodos procura identificar um padrão do Skype nos primeiros bits do *payload* e usa o teste Chi-quadrado para verificar se o conteúdo do pacote tem um padrão de dados aleatórios, indicando que o conteúdo foi criptografado.

Todas as chamadas de voz do Skype são criptografadas fim-a-fim. De acordo com informações de seu próprio *site* Internet, é usado o algoritmo Rijndael (AES) com chaves de 256 bits. A negociação das chaves simétricas AES é feita com o algoritmo RSA 1024 bits.

Os artigos apresentados também mencionam que o Skype usa como primeira opção de comunicação o protocolo UDP com uma porta escolhida aleatoriamente. Se o protocolo UDP não estiver disponível por algum motivo, como por exemplo a presença de um *firewall* que bloqueie esse tipo de tráfego, ele pode usar as portas 80 (HTTP) e 443 (HTTPS) do protocolo TCP. Adotando essa estratégia, o programa consegue funcionar com sucesso em diversas situações, já que o trânsito pelas portas HTTP é normalmente permitido em qualquer rede. A presença de um servidor *proxy* na rede também é reconhecida pelo Skype.

Além disso, o programa Skype pode redirecionar tráfego de outros clientes, ou seja, o programa, quando em execução, pode também gerar tráfego mesmo sem ação do usuário.

Qualquer computador com recursos suficientes pode começar a redirecionar tráfego de outros usuários Skype. Não se conhece um modo de configurar o programa de forma a prevenir esse redirecionamento, mas aparentemente isso não ocorre em computadores atrás de *firewalls* restritivos, já que eles mesmos vão precisar de um redirecionamento para seu tráfego. O Skype também gera tráfego periodicamente para verificar se seus pares estão ativos, descoberta de novos clientes em execução e outras operações próprias de rede *peer-to-peer*.

Devido a grande popularidade do programa Skype alcançada nos últimos anos e pela sua grande capacidade de adaptação ele foi nossa primeira opção nos testes experimentais. Em 2007, o número de usuários Skype conectados ao mesmo tempo atingiu a marca de 10 milhões. No total, o programa já contabiliza mais de 240 milhões de usuários registrados, mas como os usuários podem se registrar várias vezes, esse número não representa o número de pessoas que já usaram o Skype.

3 METODOLOGIA

Neste capítulo apresentamos a metodologia desenvolvida para detecção de anomalias em amostras de tráfego HTTP, bem como os testes e parâmetros usados. Deseja-se desenvolver um programa capaz de fazer uma rápida distinção entre tráfego Web considerado normal e tráfego Web considerado anômalo, conforme ilustrado na FIG. 3.1. O tráfego normal é composto de requisições e respostas de navegadores Web, ou seja, pessoas acessando páginas HTTP. O tráfego Web anômalo tem outros tipos de fluxos, como por exemplo, VoIP, *peer-to-peer* ou *streaming* utilizando as portas HTTP.

Em um primeiro momento, estamos interessados apenas nessa distinção entre tráfego supostamente normal ou anômalo. Uma informação desse tipo disponível para um administrador de rede em tempo real ou com um retardo de alguns segundos é importante porque poderia ser tomada uma ação imediata com relação aos fluxos anômalos. Uma detecção desse tipo que demorasse mais tempo seria interessante para uma análise do tráfego, mas não para se tomar ações imediatas, pois vários fluxos identificados poderiam não estar mais ativos.

De uma maneira similar aos sistemas de detecção de intrusão baseados em anomalias, não se deseja fazer uma distinção baseada em assinaturas de aplicativos ou no reconhecimento de detalhes específicos no tráfego de cada tipo de anomalia. Deseja-se modelar o comportamento normal do tráfego Web e comparar com o comportamento observado. Uma diferença entre esses dois conjuntos maior do que uma determinada margem de tolerância indicaria uma anomalia. Nessa situação, vão existir erros de identificação, falsos positivos e falsos negativos. Os resultados são medidos e é buscada a minimização dos erros.

Em um segundo momento, uma vez identificadas as anomalias, elas poderiam ser classificadas de acordo com seu tipo de tráfego, conforme mostra a FIG. 3.1. Nesse trabalho, embora a metodologia desenvolvida seja genérica, vamos apresentar resultados apenas relativos à detecção de fluxos VoIP no tráfego HTTP.

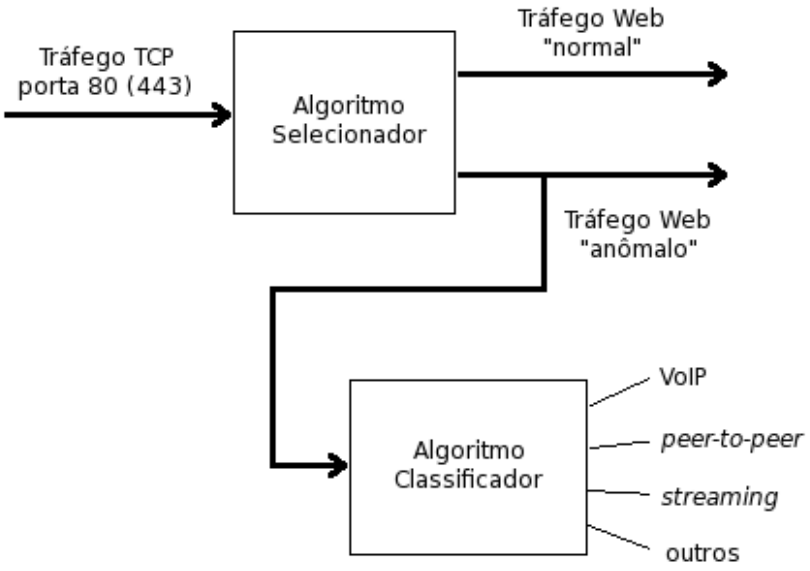

FIG. 3.1: Estrutura do sistema de detecção de anomalias proposto.

3.1 PROPOSTA DE MODELAGEM HTTP

Uma primeira etapa na caracterização do tráfego HTTP normal é escolher quais parâmetros devem ser levados em consideração. Na Seção 2.5 apresentamos alguns modelos com parâmetros de uso do protocolo HTTP que foram desenvolvidos em trabalhos anteriores (CHOI, 1999; MAH, 1997; ARLITT, 1995). Diferentemente dos trabalhos anteriores, nosso objetivo não é ter um gerador de tráfego HTTP sintético ou fazer simulações, mas sim diferenciar tráfego HTTP de não-HTTP o mais rapidamente possível. Definimos então um modelo HTTP baseado nos modelos estudados e que tenha parâmetros relativamente simples (FREIRE, 2007). Nosso modelo conta com os seguintes parâmetros:

- Tamanho da requisição;
- Tamanho da resposta;
- Intervalo de tempo entre requisições;

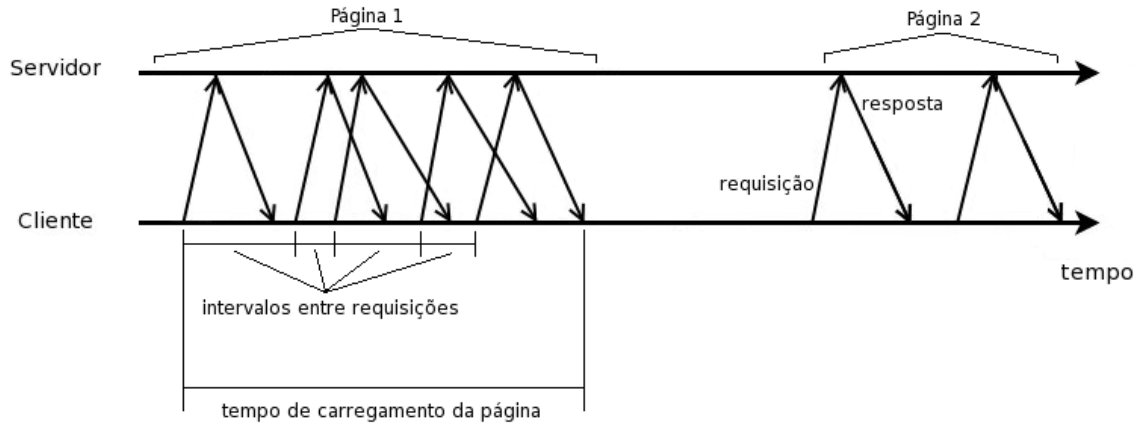

FIG. 3.2: Exemplo de mensagens HTTP em um fluxo IP.

- Número de requisições por página;
- Tempo de carregamento da página;

O tamanho da requisição consiste no tamanho em bytes da requisição HTTP enviada pelo cliente para o servidor e o tamanho da resposta consiste no tamanho em bytes da resposta HTTP enviada pelo servidor para o cliente. O intervalo de tempo entre requisições é o tempo decorrido entre o envio de duas requisições consecutivas para um mesmo servidor HTTP, desde que essas duas requisições sejam para objetos de uma mesma página HTTP. Aqui considera-se uma página HTTP o conjunto completo de objetos que formam um mesmo documento Web, que normalmente é composto de um arquivo HTML e algumas imagens. O número de requisições por página é o número de mensagens HTTP enviadas pelo cliente para um mesmo servidor relativas a uma mesma página HTTP. O tempo de carregamento da página é o tempo decorrido entre o envio da primeira requisição de uma página HTTP e a recepção da última resposta da mesma página, conforme mostrado na FIG. 3.2.

Como descrito na Seção 2.4, o protocolo HTTP tem apenas dois tipos de mensagens: requisições e respostas. Escolhemos então o tamanho das mensagens de requisição e resposta como parâmetros do nosso modelo. Outros modelos, como (MAH, 1997) e (CHOI,

1999), subdividem as mensagens de requisição e resposta em quatro grupos, separando a primeira mensagem de cada requisição (ou resposta) das demais. Isso se deve ao fato de que, tradicionalmente, as páginas Web eram formadas por um arquivo HTML único e várias imagens inseridas. Essa separação das primeiras mensagens visava colocar os arquivos HTML e os arquivos de imagens em grupos distintos. Contudo, com o desenvolvimento da Web e com a especificação 4.0 do HTML, essa característica de um único arquivo HTML em cada página Web deixou de ser comum, por isso conservamos todas as requisições (ou respostas) em um único grupo de mensagens.

Para a determinação dos outros três parâmetros de nosso modelo HTTP, é necessário identificar as fronteiras de cada página Web, que pode ter apenas uma ou inúmeras requisições. Após a definição desses limites, o número de requisições por página e o tempo de carregamento da página são facilmente calculados. O intervalo de tempo entre requisições é calculado para páginas com duas ou mais requisições, sendo a diferença de tempo entre o início de duas requisições consecutivas da mesma página, conforme mostrado na FIG. 3.2. Conforme será detalhado na Seção 3.3.2, o tamanho da requisição, o tamanho da resposta e o intervalo de tempo entre requisições são parâmetros usados para a geração de distribuições empíricas, o número de requisições por página é usado para filtrar fluxos com poucas requisições e o tempo de carregamento da página é usado na combinação dos resultados quando um fluxo IP possuir várias páginas HTTP.

3.2 TESTES DE ADERÊNCIA

No caso em que não se sabe ao certo a distribuição de um conjunto de dados, pode-se usar um teste de aderência para verificar se uma dada distribuição pode ser adequada como um modelo da população. Neste trabalho foram escolhidos duas métricas retiradas de dois testes de aderência: o valor χ^2 do teste Chi-quadrado (COCHRAN, 1952) e a distância D do teste Kolmogorov-Smirnov (MASSEY, 1951). Esses testes já tinham sido usados em trabalhos anteriores na área de detecção de intrusão (YE, 2001a; ESTÉVEZ-TAPIADOR, 2004) ou para verificar a presença de pacotes com dados aparentemente aleatórios em uma detecção Skype (BONFIGLIO, 2007).

Na versão completa de cada um desses testes estatísticos, os valores χ^2 ou D são

usados para se aceitar ou rejeitar uma hipótese inicial com um certo grau de significância de acordo com distribuições conhecidas. O grau de significância é a probabilidade da hipótese inicial ser rejeitada quanto ela é verdadeira, ou seja, a probabilidade de um erro tipo I (falso positivo). Neste trabalho os testes não foram usados na forma completa, pois os valores χ^2 ou D calculados são usados diretamente na comparação com valor limiar para decidir se um dado fluxo é normal ou não. Essa solução produz uma maior simplicidade e flexibilidade para nosso programa, uma vez que é necessário apenas uma mudança nos limiares de detecção para uma classificação mais conservadora (isto é, com poucos falsos positivos) ou uma classificação mais relaxada (isto é, com uma alta sensibilidade).

3.2.1 CHI-QUADRADO

O teste χ^2 avalia uma hipótese inicial de que as freqüências observadas de eventos independentes seguem uma certa distribuição. Vamos supor que temos n observações de uma população classificada em k classes mutuamente exclusivas e existe alguma teoria ou hipótese que diz que uma observação pertence a classe i com probabilidade p_i ($i = 1, \ldots, k$), então o número de eventos esperados na classe i é $E_i = np_i$. Sendo O_i o número de eventos realmente observados na classe i, a estatística χ^2 é a soma dada por

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}. \tag{3.1}$$

Um grande valor dessa soma indica que dificilmente os valores O_i são derivados da população representada pelos valores E_i.

Existem diferentes variações desse teste, a forma mais completa compara o valor χ^2 com uma tabela de valores da distribuição χ^2 de acordo com o número de graus de liberdade do sistema. Em outras palavras, a hipótese inicial é rejeitada se $\chi^2 > \chi^2_{(\alpha,k)}$, onde $\chi^2_{(\alpha,k)}$ é o valor da distribuição com grau de significância α e k graus de liberdade.

3.2.2 KOLMOGOROV-SMIRNOV

O teste Kolmogorov-Smirnov (MASSEY, 1951) verifica se uma determinada amostra provem de uma população com uma distribuição determinada. Ele é aplicado a distribuições contínuas de uma única dimensão, que aqui vamos chamar de $F_0(x)$ e $S_N(x)$.

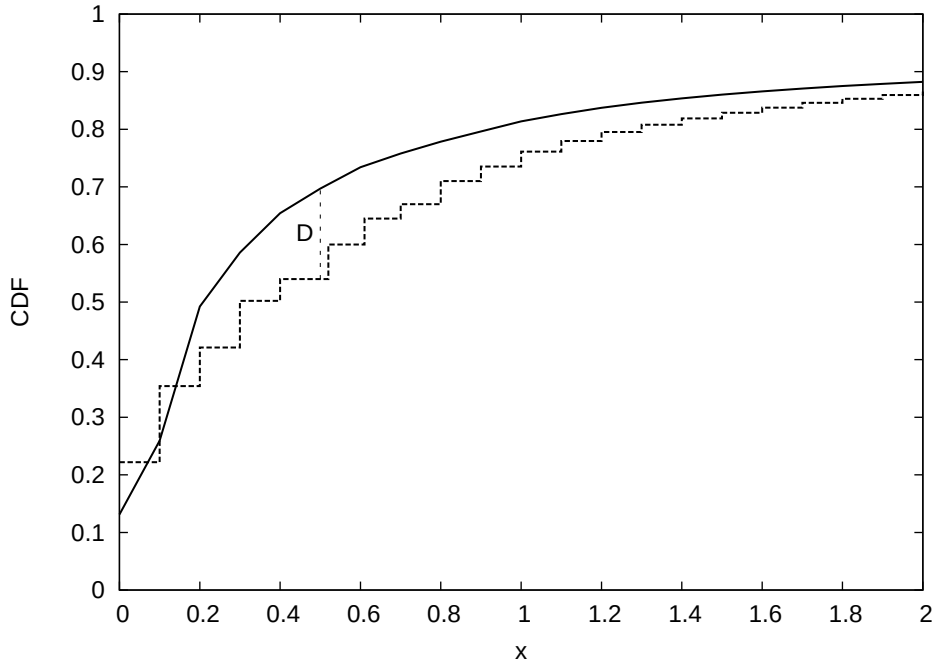

FIG. 3.3: Representação gráfica da distância Kolmogorov-Smirnov.

$F_0(x)$ seria a função de distribuição cumulativa que acredita-se que pode representar a população e $S_N(x)$ a função cumulativa construída a partir de N observações. $S_N(x)$ é uma função do tipo degrau, pois para cada x, sendo c o número de observações com valor menor que x, temos que $S_N(x) = c/N$. De maneira similar, para um dado x, $F_0(x)$ representa a fração de indivíduos na população que tem um valor menor ou igual a x. O valor Kolmogorov-Smirnov é a máxima distância entre essas duas distribuições dada por

$$D = \max\big(|S_N(x) - F_0(x)|\big). \tag{3.2}$$

O valor D também pode ser determinado graficamente, medindo-se a maior distância entre as distribuições, conforme ilustrado na FIG. 3.3. Da mesma forma, a versão completa do teste compara o valor D calculado com uma tabela de valores críticos de acordo com o tamanho da amostra N e grau de significância α.

3.3 PROCESSO DE DETECÇÃO

O processo de detecção aqui apresentado é composto de duas etapas. Primeiramente, na etapa de caracterização, o modelo definido na Seção 3.1 deve ser povoado, ou seja,

deve-se procurar medir os valores comuns para cada um dos cinco parâmetros listados. No nosso caso, geraremos uma distribuição empírica a partir de dados reais capturados de enlaces Internet para cada um dos parâmetros utilizados. Após isso, usaremos os testes de aderência para verificar se um novo conjunto de dados gerado a partir de um mesmo fluxo HTTP é semelhante ou não às distribuições definidas.

3.3.1 CARACTERIZAÇÃO

A primeira etapa do processo de detecção pode ser vista como um treinamento, ou uma caracterização do comportamento normal do tráfego HTTP. Alguns trabalhos como (BARFORD, 1998; CHOI, 1999) procuraram modelar parâmetros HTTP ajustando os valores observados para distribuições conhecidas, enquanto outros trabalhos como (ABRAHAMSSON, 2000; MAH, 1997) usaram distribuições empíricas. Pela falta de um consenso sobre qual distribuição seria mais adequada para cada parâmetro, aqui também decidiu-se usar distribuições empíricas.

A maneira mais direta para a geração das distribuições empíricas é capturar uma quantidade representativa de pacotes HTTP, calcular os parâmetros para cada fluxo e combinar os resultados em uma distribuição de valores. Para a captura dos pacotes, optou-se por usar o `tcpdump`[4], um conhecido programa com diversas funcionalidades. Neste caso ele será usado para filtrar e capturar pacotes HTTP para arquivos binários (no padrão da biblioteca `libpcap`).

Foi desenvolvido um programa chamado `ckfflow` para ler os arquivos de captura e calcular os parâmetros do modelo Web definido na Seção 3.1. O programa desenvolvido foi baseado em outro já existente chamado `tcpflow` [5], que é um programa com licença GPL capaz de ler arquivos de captura `tcpdump`/`libpcap` e separar o conteúdo de cada fluxo IP em um arquivo diferente. O `ckfflow`, além de separar os diversos fluxos IP presentes no arquivo de captura, determina os limites das páginas HTTP presentes em cada fluxo e calcula os parâmetros mencionados para cada página HTTP. As páginas presentes em

[4]Programa desenvolvido por Van Jacobson, Craig Leres e Steven McCanne. Disponível no endereço http://www.tcpdump.org/ [2007].

[5]Programa desenvolvido por Jeremy Elson. Disponível na página pessoal do autor no endereço http://www.circlemud.org/jelson/software/tcpflow/ [2007].

cada fluxo são identificadas com base no tempo de inatividade do fluxo. Um requisição enviada mais de 1 segundo após a última resposta recebida é considerada uma nova página HTTP. Com intervalos menores que 1 segundos, as requisições são consideradas da mesma página.

Nessa fase, a captura feita pelo `tcpdump` deve ser de pacotes TCP/IP completos, ou seja, com cabeçalho IP, cabeçalho TCP e todo o conteúdo de dados. Essa captura completa é necessária porque nosso programa deve ler informações do cabeçalho HTTP para determinar com precisão o tipo de mensagem HTTP (requisição ou resposta) e os limites de cada mensagem. Como visto na Seção 2.4, as informações do cabeçalho HTTP são transmitidas em formato texto, sem especificação de tamanho, sem um tamanho máximo definido e separado do conteúdo da mensagem HTTP por uma linha em branco. Desta forma a captura completa do conteúdo TCP é necessária para se ter a garantia de que todos os cabeçalhos HTTP serão capturados.

O programa `ckfflow` também é responsável pela filtragem dos dados não reconhecidos como HTTP. Nesta etapa, deseja-se uma garantia de que todos os dados levados em consideração na geração das distribuições empíricas são de verdadeiras mensagens HTTP. Se um determinado fluxo não tiver um cabeçalho HTTP mínimo reconhecido pelo programa, o fluxo deve ser descartado do restante da análise.

Depois de serem analisados e computados os parâmetros para todos os fluxos presentes no arquivo de captura, os resultados são combinados para a geração das distribuições empíricas. O `ckfflow` deve gerar como saída cinco arquivos texto, um para cada parâmetro listado, com a listagem de todos os valores calculados. A partir de cada listagem de valores deve ser gerada uma distribuição empírica para a etapa de detecção. Neste caso, cada listagem é ordenada e dividida em vários intervalos e o número de resultados presentes no arquivo texto para cada intervalo é contado, gerando-se uma distribuição empírica. Cada intervalo é considerado uma classe i no cálculo do valor χ^2 e dependendo do tamanho das classes, o valor χ^2 pode variar. Como a escolha dos tamanhos ideais depende da distribuição, a definição do tamanho dos intervalos foi feita empiricamente, levando-se em consideração que o tamanho dos intervalos e a freqüência esperada em cada intervalo não precisavam ser constantes.

Para a geração da função de distribuição cumulativa (CDF), a listagem de valores foi ordenada, dividida empiricamente em intervalos de igual tamanho e foi calculada a fração de resultados presentes até cada intervalo.

O procedimento descrito nesta seção deve ser realizado uma vez no início da análise. As distribuições geradas vão necessitar de atualização devido a mudanças de comportamento no tráfego Web. Por exemplo, o trabalho (MAH, 1997), que analisou dados colhidos em 1995, apresentou o tamanho médio das respostas HTTP como algo em torno de 1,5 a 2,0 KB. Nossos dados capturados, mostrados na Seção 4.1, vão indicar um tamanho médio bem superior. Mesmo com essa constante evolução nos padrões de tráfego HTTP, vamos considerar que para um determinado intervalo de tempo, o comportamento do tráfego se mantém constante. No momento não estamos interessados em estudar o período de validade dessas distribuições empíricas, mas vamos assumir que elas permanecem válidas pelo menos até o final da captura dos dados para teste de detecção, o que normalmente ocorreu algumas semanas após cada captura de treinamento.

3.3.2 DETECÇÃO

A segunda etapa do processo representa a detecção propriamente dita. Nosso objetivo é comparar fluxos novos com as distribuições empíricas determinadas na etapa anterior. Novamente, a maneira usual de se conseguir isso é capturar pacotes HTTP e analisar os dados capturados. O programa `tcpdump` também será usado para essa captura, mas com a diferença de que agora apenas os cabeçalhos TCP/IP devem ser capturados. Na verdade o programa `tcpdump` não analiza os campos dos cabeçalhos IP ou TCP para determinar com precisão onde termina cada um deles. Por padrão, ele faz uma captura dos primeiros 96 bytes de cada pacote, o que é suficiente em praticamente todos os casos para capturar os cabeçalhos IP e TCP.

Nesta etapa são capturados apenas os cabeçalhos TCP/IP porque não se deseja observar o conteúdo dos pacotes. Na fase anterior o conteúdo foi observado para se ter uma garantia de que todos os fluxos levados em consideração eram HTTP e para se determinar com precisão seus parâmetros. Na detecção, uma análise desse tipo ia ser mais bem demorada, ia colocar em discussão questões sobre a privacidade dos dados e não seria

possível a análise do tráfego HTTPS, entre outros fatores.

Foi desenvolvido para a etapa de detecção um outro programa, chamado `dkfflow`, com a finalidade de ler os arquivos de captura e calcular cada parâmetro do modelo HTTP. Esse programa se diferencia do programa `ckfflow` porque agora o cálculo dos parâmetros é feita sem a visualização do conteúdo dos pacotes. O programa `dkfflow` supõe que todos os fluxos presentes são de tráfego HTTP e analisa todos da mesma forma, mesmo sem a garantia de que todos sejam HTTP. Dessa maneira, a privacidade dos dados é mantida e torna-se possível a análise de tráfego HTTPS ou de tráfego HTTP que tenha um conteúdo de pacote criptografado.

O programa `dkfflow` deve procurar definir onde começa e onde termina cada página Web não mais pela observação do cabeçalho HTTP, mas sim baseado no tempo de inatividade do fluxo (o intervalo de tempo entre a recepção de um pacote resposta e o envio da próxima requisição). Para a definição do tamanho da requisição ou da resposta, usa-se o tamanho máximo de uma unidade de transmissão IP (MTU). O tamanho máximo possível para um pacote IPv4 é 65.535 bytes, mas quando se usa enlaces de rede tipo Ethernet, o tamanho máximo permitido normalmente é 1.500 bytes. Assim, considera-se que os pacotes que tem um tamanho igual (ou muito próximo) ao tamanho máximo permitido (MTU) representam o mesmo objeto HTTP, desde que não exista um grande intervalo de tempo entre eles. Os pacotes menores representariam a parte final de um objeto HTTP.

Esses procedimentos podem inserir alguns erros de interpretação em fluxos HTTP verdadeiros, mas espera-se que esses casos fiquem dentro de uma margem de tolerância razoável. Mesmo no caso de um fluxo VoIP, que não é formado por mensagens de requisição e respostas nem é agrupado em páginas, o programa irá interpretá-lo como se fosse um fluxo HTTP. O resultado dessa interpretação pode ser visto como o conjunto de características que um fluxo HTTP deveria ter para que seu tráfego seja similar ao fluxo VoIP em questão. Para que a detecção tenha bons resultados, basta que essa caracterização de fluxos VoIP como HTTP resulte em algo bem diferente dos fluxos HTTP normais.

Para cada página Web, os cinco parâmetros são calculados e recebem tratamentos diferentes. O número de requisições por página e o tempo de carregamento da página são supostamente correlacionados e produzem um único valor para cada página Web. O

número de requisições por página foi usado como um filtro para remover da análise os fluxos com poucas requisições. Como ele produz um único valor para cada página Web, não seria indicado para os testes de aderência, que esperam como entrada uma lista de valores.

Os outros três parâmetros (tamanho da requisição, tamanho da resposta e intervalo de tempo entre requisições) produzem uma lista de valores para cada página Web, podendo ser usados nas Equações (3.1) e (3.2). O número de elementos dessa lista aumenta a medida que o número de pares requisição/resposta aumenta, então os fluxos com poucas requisições não são levados em consideração para que não sejam geradas listas com poucos valores. Assume-se que as listas com poucos valores, ao serem usadas nas Equações (3.1) e (3.2), podem levar a desvios no resultado da expressão.

Cada um dos parâmetros é avaliado separadamente, gerando um arquivo texto de saída com os valores χ^2 e D calculados e o programa `dkfflow` termina sua execução. Para a comparação dos valores χ^2 calculados e a identificação dos fluxos anômalos, foi desenvolvido um programa chamado `testchi`. Da mesma forma, o programa `testks` compara os valores D calculados para a identificação dos fluxos anômalos. Cada programa vai comparar os valores calculados (χ^2 ou D) com limiares de detecção para a classificação do pedaço de fluxo em questão como sendo HTTP normal ou anômalo. Para a classificação do fluxo como um todo, os resultados de cada pedaço (página) são combinados em uma soma ponderada, usando-se o tempo de carregamento da página como peso na soma ponderada. Se a maior parte do fluxo for considerada uma anomalia, o fluxo todo é considerado uma anomalia. No caso em que um fluxo for dividido em apenas um pedaço (página), a classificação dele já seria a classificação do fluxo.

3.4 PROGRAMAS DESENVOLVIDOS

Os principais programas desenvolvidos durante o decorrer do trabalho estão listados a seguir. Todos eles foram desenvolvidos em linguagem C e compilados em ambiente Linux usando-se o compilador `gcc`. Embora as versões dos programas sejam funcionais, elas não foram escritas de uma maneira otimizada para uso em situações reais, elas foram desenvolvidas com o objetivo de se avaliar a metodologia desenvolvida, sem grandes preocupações

com facilidade de uso, tratamento de erros ou portabilidade, entre outros.

`akfflow`: Desenvolvido para anonimizar os arquivos de captura. O programa consegue ler um arquivo de captura no formato `tcpdump/libpcap` e gerar uma nova versão com os endereços IP de origem e destino de todos os pacotes alterados. Esses dados foram alterados usando-se uma função hash (MD5), de forma que endereços IP iguais sejam mapeados para um mesmo valor.

`ckfflow`: Programa desenvolvido para realizar a caracterização dos fluxos HTTP no processo de geração das distribuições empíricas. Seu funcionamento é detalhado na Seção 3.3.1.

`dkfflow`: Programa desenvolvido para realizar o cálculo dos valores χ^2 e D para fluxos presentes em um arquivo de captura. Seu funcionamento é detalhado na Seção 3.3.2.

`testchi`: Programa que testa os valores χ^2 comparando-os com os limiares de detecção para uma classificação do fluxo como um todo em tráfego normal ou anômalo. Seu funcionamento é detalhado na Seção 3.3.2.

`testks`: Programa que testa os valores da distância D Kolmogorov-Smirnov e compara esses valores com limiares de detecção para uma classificação do fluxo como um todo em tráfego normal ou anômalo. Seu funcionamento é detalhado na Seção 3.3.2.

4 AVALIAÇÃO DE DESEMPENHO

Conforme indicado no Capítulo 3, necessitamos de diversos conjuntos de dados HTTP para fazermos uma avaliação do sistema. O primeiro conjunto seriam dados de treinamento, para a construção de uma distribuição empírica para cada parâmetro do nosso modelo. Na Seção 4.1 apresentamos os diversos conjuntos de dados usados para treinamento e as distribuições geradas a partir deles.

Um segundo conjunto de dados foi gerado com o objetivo de se ter uma avaliação preliminar dos atributos do sistema, ou seja, dos parâmetros usados no modelo HTTP e das métricas escolhidas. Para tanto, foi realizada uma comparação entre um conjunto de dados HTTP e outro com tráfego VoIP. Os resultados dessa avaliação estão apresentados na Seção 4.2.

A partir dos resultados promissores obtidos no segundo conjunto de dados, partimos para uma situação de detecção real, capturando vários conjuntos de dados com tráfego VoIP misturado com HTTP para que nosso sistema identifique os fluxos VoIP presentes. Os resultados obtidos estão na Seção 4.3.

Por fim, reunimos um conjunto de dados para avaliar a eficiência do nosso sistema de detecção em uma situação de tempo real, onde se deseja uma correta classificação dos fluxos em um intervalo de tempo de alguns segundos. A Seção 4.4 mostra os resultados obtidos nessa última avaliação.

4.1 GERAÇÃO DAS DISTRIBUIÇÕES EMPÍRICAS

Uma parte de grande importância para o nosso estudo consiste em achar uma fonte de dados HTTP representativa que possa ser usada para a geração das distribuições empíricas. Como estamos interessados em diferenciar tráfego HTTP normal do anômalo, devemos assegurar que os conjuntos de dados usados para treinamento não contenham as anomalias que estaremos procurando detectar. Por esta razão, nesta fase é realizada uma captura de pacotes TCP/IP completos, com cabeçalho e todo o conteúdo.

TAB. 4.1: Relação de tráfego Web capturado para treinamento.

Nome	Período	Volume médio diário
ISP-T1	18-20 Junho 2007	45 GB
ACD-T1	14-17 Agosto 2007	35 GB
ISP-T2	14-16 Agosto 2007	50 GB
ISP-T3	18-20 Outubro 2007	60 GB

Foi capturado tráfego real de dois tipos de instituições conectadas à Internet: um provedor de acesso Internet comercial, que chamaremos de ISP e uma instituição acadêmica, que será denominada ACD. A TAB. 4.1 apresenta a listagens de todas as capturas realizadas para essa etapa. As capturas ISP-T1, ISP-T2 e ISP-T3 foram feitas com intervalos de dois meses a partir de um mesmo provedor de acesso, localizado na cidade de Niterói, RJ. Por questões técnicas, a captura não foi feita diretamente do enlace Internet principal do provedor de acesso, mas a partir de dois enlaces secundários que juntos eram responsáveis por praticamente todo o tráfego do enlace Internet principal. O provedor de acesso possuía na época cerca de 3 mil clientes ativos, e cada um dos dois enlaces atendia a uma fração desses clientes. O volume maior de dados colhidos na captura ISP-T3 sugere que a maior parte dos clientes trafegava no enlace observado pela captura ISP-T3.

A captura ACD-T1 foi feita a partir do enlace principal da Internet de uma instituição acadêmica localizada na cidade de Petrópolis, RJ e que possuía na época cerca de 300 usuários. Por conta do menor número de usuários em relação ao provedor de acesso, o volume de dados colhidos na captura ACD-T1 foi menor em comparação com as demais capturas.

O volume médio diário listado na TAB. 4.1 é relativo apenas ao volume de tráfego HTTP capturado, ou seja, o tráfego da porta 80/TCP, que é filtrado e capturado usando-se o programa `tcpdump`. O tráfego total em cada enlace tem um volume maior, como pode ser observado na FIG. 4.1, que representa a variação de volume total de tráfego no enlace durante a captura ISP-T1. Para realizar essas capturas foi utilizado um computador ligado a uma porta Ethernet que estava espelhando o tráfego de saída. A quantidade de pacotes descartados pelo programa `tcpdump` durante a captura foi considerada desprezível, ou seja, a configuração de captura mostrou-se compatível com o volume de dados do enlace.

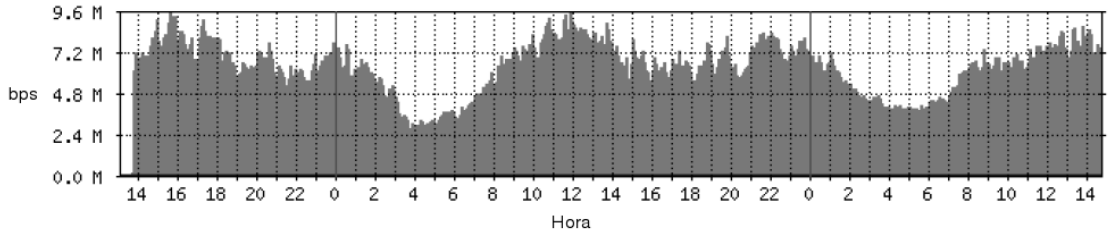

FIG. 4.1: Volume de tráfego em bits por segundo observado durante ISP-T1.

TAB. 4.2: Detalhes sobre os parâmetros calculadas a partir de ISP-T1.

Parâmetro	Número de Observações	Valor Médio
Tamanho das requisições Web (bytes)	6.886.175	616,718
Tamanho das respostas Web (bytes)	3.261.266	8.538,294
Número de requisições por página	2.077.618	2,059
Tempo de carregamento da página (s)	2.077.618	5,720
Intervalo de tempo entre requisições Web (s)	1.153.552	0,674

O tráfego em cada enlace apresenta variações de acordo com a hora do dia e o dia da semana. Buscando minimizar a influência dessas flutuações de horário, cada uma das capturas ISP-T1, ISP-T2 e ISP-T3 teve uma duração de 48 horas e a captura ACD-T1 teve uma duração de 72 horas, ou seja, cada uma delas terminou na mesma hora que tinha começado alguns dias antes. Além disso, foram escolhidos preferencialmente dias úteis para a captura dos dados. Por exemplo, a captura ISP-T1, mostrada na FIG. 4.1, começou às 14 horas do dia 18 de junho de 2007 (segunda-feira) e terminou às 14 horas do dia 20 de junho de 2007 (quarta-feira).

Antes de usar o programa `ckfflow` mencionado na Seção 3.3.1 para analisar os arquivos de captura, buscou-se anonimizar os dados relativos aos endereços IP. Para tanto foi utilizado o programa `akfflow` para mudar de forma automática todos os endereços IP de origem e destino dos pacotes.

A TAB. 4.2 apresenta como referência o número de observações e o valor médio calculado para cada parâmetro do modelo a partir da captura ISP-T1. Com os dados colhidos durante as 48h de captura, nosso programa identificou um total de 6.886.175 mensagens de requisição HTTP e 3.261.266 mensagens de resposta. Nessa operação, foram considera-

das apenas as mensagens de requisição do tipo GET ou POST e as mensagens de resposta com código 200 (OK). A grande diferença entre esses dois números pode ser explicada pela presença de mensagens de resposta com outros códigos que não o 200 e pela presença de muitas mensagens de requisição que não tiveram resposta.

É possível que algumas requisições HTTP presentes no arquivo de captura tenham sido feitas por clientes automatizados ou Web *spiders*. Esse tipo de processo automático poderia ser identificado por um grande número de requisições em um pequeno intervalo de tempo ou por apresentar requisições de forma periódica, em intervalos de tempo aproximadamente iguais. Contudo, esses tipos de requisições não foram explicitamente filtradas dos nossos arquivos de captura.

As diferenças existentes entre as versões do protocolo HTTP também podem influenciar nos resultados obtidos. Esperava-se fazer duas análises, uma com as mensagens HTTP/1.0 e outra com as mensagens HTTP versão 1.1. Entretanto, a porcentagem observada de mensagens HTTP/1.0 foi menor que 5% em todas as capturas realizadas. Desta forma, toda a análise foi baseada apenas na versão HTTP/1.1 e as mensagens HTTP/1.0 não foram levadas em consideração na formação das distribuições.

Como pode ser visto na TAB. 4.2, o número de páginas HTTP identificadas pelo programa foi de 2.077.618 para a captura ISP-T1. Cada página é formada por um ou mais pares de mensagens requisição/resposta, ou seja, existiram pouco mais de 2 milhões de páginas HTTP com pelo menos uma requisição e uma resposta de acordo com nossos requisitos. Foi observado também um grande número de páginas com apenas 1 par requisição/resposta. Caso a maior parte das mensagens do arquivo fosse da versão HTTP/1.0, isso seria um resultado esperado, já que o protocolo HTTP/1.0 normalmente tem uma requisição por conexão. No nosso caso, porém, as mensagens HTTP/1.0 não foram levadas em consideração pelo programa. Por fim, o intervalo de tempo entre requisições Web teve apenas 1.153.552 valores computados. Logicamente, para o cálculo desse parâmetro apenas foram consideradas páginas HTTP com duas ou mais requisições de acordo com os requisitos apresentados.

As distribuições geradas pelos três parâmetros que foram escolhidas para serem avaliadas são apresentadas na FIG. 4.2, FIG. 4.3 e FIG. 4.4. As figuras já mostram o resultado

obtido para cada uma das as capturas listadas na TAB. 4.1. Observa-se que, de maneira geral, as capturas ISP-T1, ISP-T2 e ISP-T3 tiveram resultados bastante próximos.

A FIG. 4.2 apresenta a função de distribuição cumulativa (CDF) para o tamanho da requisição observado em cada captura. Enquanto as capturas ISP apresentaram uma distribuição de valores similar, a captura ACD-T1 apresentou uma curva levemente inferior. Mas para todas as capturas, o tamanho das requisições dificilmente foi superior a 1.500 bytes. As mensagens maiores que esse valor representaram menos de 0,8% do volume total de requisições.

Na FIG. 4.3 temos a função de distribuição cumulativa para o tamanho da resposta. Novamente, as capturas ISP foram bem semelhantes e a captura ACD-T1 apresentou uma distribuição ligeiramente inferior para valores até 40.000 bytes. Para todas as capturas, o número de respostas Web maiores de 100.000 bytes foi menor que 1% do total de respostas, mas esse pequeno grupo de mensagens foi responsável por algo em torno de 30 a 40% do volume das respostas.

A FIG. 4.4 mostra os resultados para o intervalo de tempo entre requisições. Ele é uma métrica para páginas Web com duas ou mais requisições, já que representa o intervalo de tempo entre requisições consecutivas de uma página. Pode-se observar que novamente as capturas ISP tiveram um resultado bastante próximo e a captura ACD foi levemente diferente.

Das capturas listadas na TAB. 4.1, foi conservado apenas um arquivo texto com os valores dos pontos de cada distribuição mostrada na FIG. 4.2, FIG. 4.3 e FIG. 4.4. Todos os outros dados foram descartados, pois o programa usado na etapa de detecção necessita apenas do conjunto de pontos relativo a cada gráfico.

A partir da semelhança dos resultados obtidos a partir das capturas ISP, gravadas em intervalos de tempo separados por alguns meses, pode-se supor que os dados ISP-T1, colhidos em junho, continuaram válidos até outubro. Da mesma forma, os dados ISP-T2, colhidos em agosto, continuariam válidos em outubro e os três conjuntos ISP-T1, ISP-T2 e ISP-T3 seriam intercambiáveis para a etapa de detecção. Com relação aos dados ACD, seria necessário uma avaliação melhor antes de se supor que seus resultados seriam intercambiáveis com um dos conjuntos ISP.

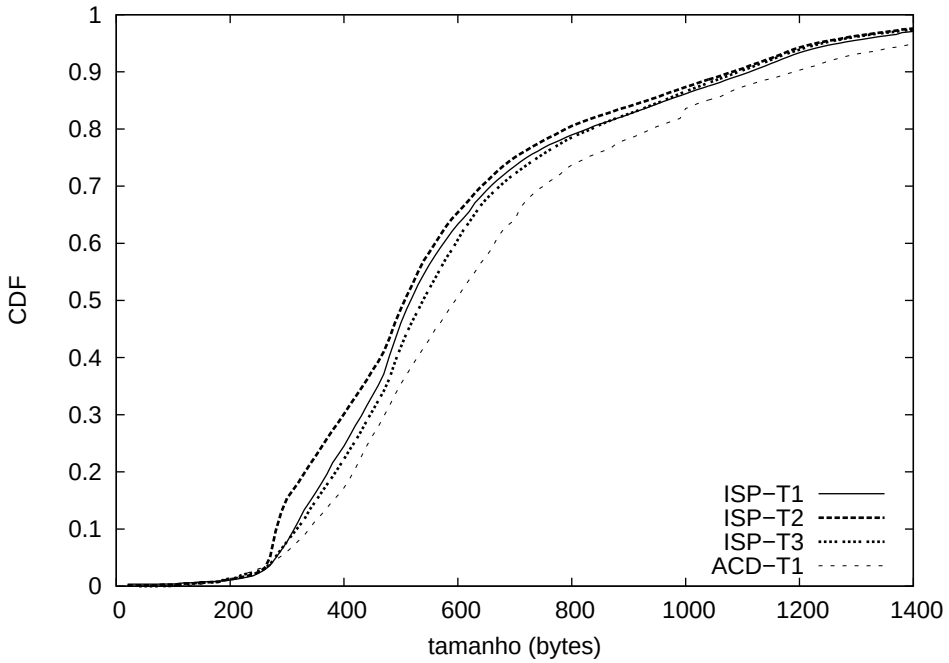

FIG. 4.2: CDF para o tamanho das requisições Web.

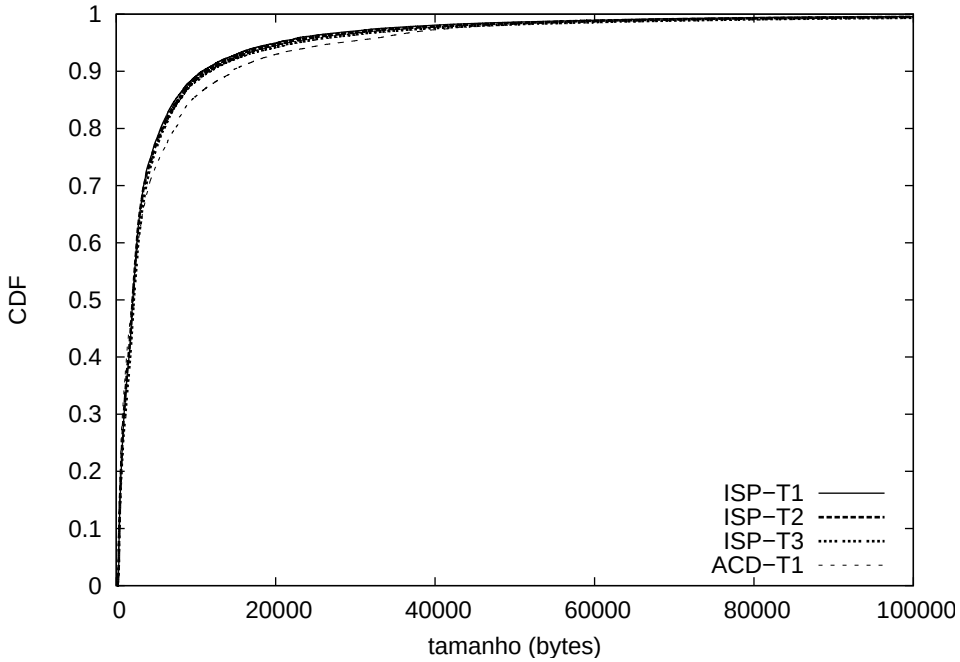

FIG. 4.3: CDF para o tamanho das respostas Web.

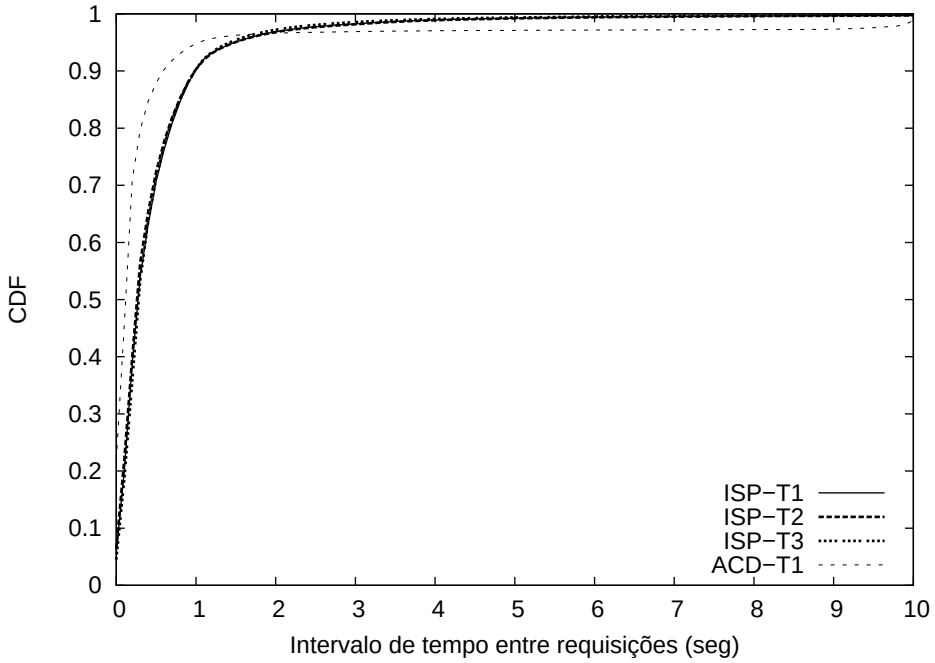

FIG. 4.4: CDF para o intervalo de tempo entre requisições Web.

4.2 AVALIAÇÃO DAS MÉTRICAS

Antes de iniciar uma detecção real, ou seja, testar um conjunto de dados onde existam fluxos VoIP misturados com fluxos HTTP para que nosso programa identifique os fluxos VoIP, foi realizada uma avaliação preliminar (FREIRE, 2007). Nesta avaliação, foram gerados dois conjuntos distintos, um chamado conjunto Skype com fluxos VoIP usando portas HTTP e outro chamado conjunto Web com tráfego HTTP.

O conjunto Web foi capturado do mesmo provedor de acesso usado na Seção 4.1 e teve uma duração de aproximadamente 5 horas. Diferentemente dos casos mencionadas anteriormente, agora foi realizada uma captura apenas dos cabeçalhos dos pacotes TCP/IP que trafegavam pelas portas Web. Não foi realizado nenhum tipo de filtragem nesse conjunto de dados, ou seja, não foram filtradas mensagens HTTP versão 1.0, respostas com código de retorno diferente de 200 ou requisições com métodos diferentes de GET ou POST.

Para o conjunto Web, não foi realizada uma filtragem de tráfego Skype pois o programa só usa as portas Web quando o protocolo UDP não está disponível. Como o provedor de acesso não possui nenhum tipo de restrição de protocolos, assume-se que a quantidade de

tráfego Skype no conjunto Web é desprezível.

O conjunto Skype foi gerado por um computador com o programa Skype instalado, computador este que estava atrás de um *firewall* que permitia o ingresso de tráfego apenas das portas HTTP (80 e 443/TCP) e DNS (53/UDP). Em um período de aproximadamente 5 horas, foram realizadas um total de 40 chamadas Skype com diferentes durações, conforme indicado na TAB. 4.3. Não foi realizado nenhum tipo de acesso HTTP durante essa captura, ou seja, o arquivo é exclusivo de chamadas VoIP e outros tráfegos do Skype.

TAB. 4.3: Duração das chamadas de teste Skype.

Duração (min)	Número de chamadas
0–1	12
1–5	9
5–10	9
10–15	5
15–25	5

As 5 horas de tráfego HTTP capturado geraram um volume de dados que vamos chamar de conjunto Web. Esse conjunto foi submetido a uma versão preliminar do nosso programa `dkfflow`, que terminava sua execução quando todos os valores de χ^2 e da distância D tivessem sido calculados, de acordo com a EQ. 3.1 e EQ. 3.2, respectivamente. Da mesma forma, nosso tráfego capturado Skype, chamado conjunto Skype, também foi submetido ao mesmo programa para o cálculo dos valores χ^2 e D. As funções de distribuição cumulativa (CDF) relativas aos valores χ^2 calculados para o tamanho da requisição, tamanho da resposta e intervalo de tempo entre requisições estão apresentadas na FIG. 4.5, FIG. 4.6 e FIG. 4.7, respectivamente. De maneira similar, a FIG. 4.8, FIG. 4.9 e FIG. 4.10 mostram as funções de distribuição cumulativa para os valores D calculados.

Observa-se a partir da FIG. 4.5 que nosso conjunto Skype produz valores χ^2 normalmente muito maiores que os obtidos pelo nosso conjunto Web para os tamanhos da requisição. Um comportamento similar é observado também para os tamanhos das respostas e o intervalo de tempo entre requisições, conforme observado na FIG. 4.6 e FIG. 4.7. Os resultados sugerem que pode ser escolhido um valor limiar para χ^2 de forma a classificar um fluxo desconhecido em um dos tipos apresentados. A diferença observada entre os resultados dos conjuntos também confirma nossa hipótese que a quantidade de tráfego

Skype na nossa captura Web é insignificante.

Na FIG. 4.8, temos a função de distribuição cumulativa para os valores D relativos ao tamanho das respostas. Neste caso, observa-se que nosso conjunto Web ainda pode ser claramente diferenciado do conjunto Skype selecionando-se um valor limiar apropriado. Na FIG. 4.9, uma característica semelhante é observada, mas a FIG. 4.10 já não mostra uma separação clara entre os dois conjuntos. Em todo caso, uma visão combinada dos três parâmetros é suficiente para diferenciar tráfego Skype de tráfego WWW.

4.3 DETECÇÃO

Os resultados da Seção 4.2 indicaram que os parâmetros escolhidos para a modelagem HTTP e a metodologia proposta poderiam ser usados para um sistema de detecção automatizado de fluxos Skype em tráfego HTTP. Agora vamos procurar avaliar a qualidade dessa detecção, usando arquivos de captura com fluxos HTTP e Skype misturados e medindo a quantidade de fluxos corretamente identificados e falsos positivos (FREIRE, 2008). Com esse objetivo, foram realizadas três capturas em períodos distintos, conforme descrito na TAB. 4.4. Todas elas foram realizadas a partir do mesmo provedor de acesso Internet descrito na Seção 4.1, que fornece para seus clientes endereços IP válidos e dinâmicos, para diferentes conexões.

TAB. 4.4: Relação de tráfego Web capturado para detecção.

Nome	Data	Duração	Número de fluxos VoIP
ISP-D1	23 Julho 2007	8h	80
ISP-D2	22/23 Agosto 2007	16h	85
ISP-D3	24/25 Outubro 2007	14h	115

Em uma primeira fase, optou-se por detectar apenas o tráfego gerado de chamadas de voz realizadas pelo programa Skype utilizando as portas TCP de HTTP/HTTPS. Supõe-se que não existe tráfego Skype nas portas Web além daquele inserido para avaliação pois o programa Skype escolhe preferencialmente o protocolo UDP para ligações. Como não existem portas bloqueadas, servidores de *proxy* ou *firewalls* no caminho dos clientes do provedor de acesso para a Internet, caso algum cliente faça uma ligação usando Skype no

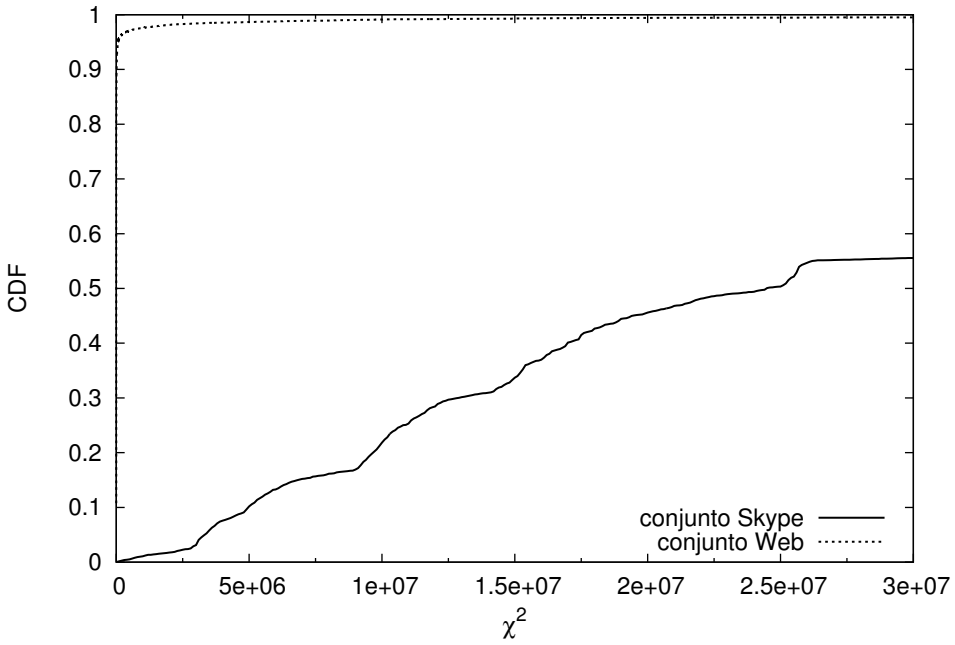

FIG. 4.5: CDF para χ^2 calculado a partir do tamanho da requisição.

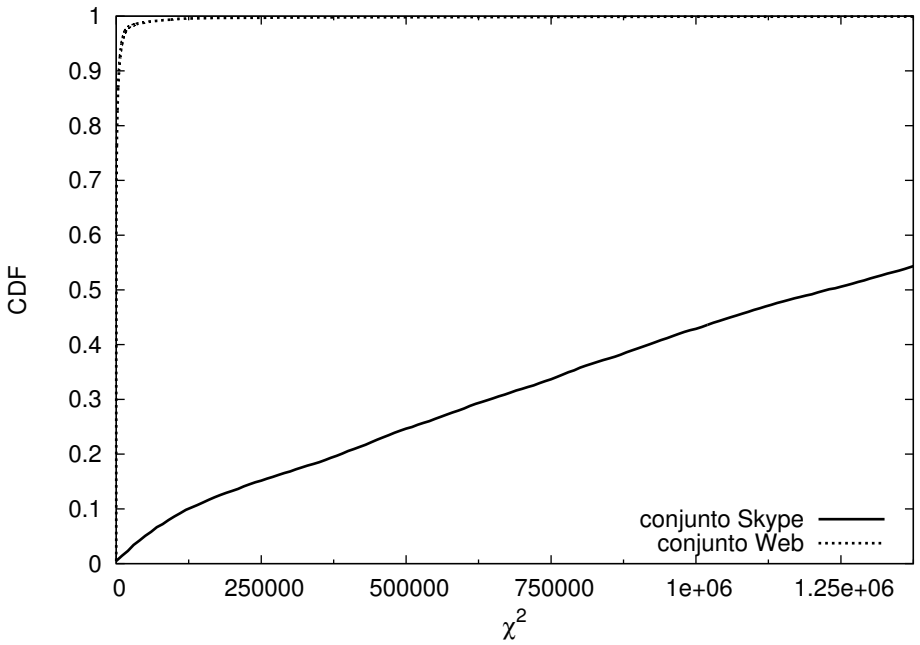

FIG. 4.6: CDF para χ^2 calculado a partir do tamanho da resposta.

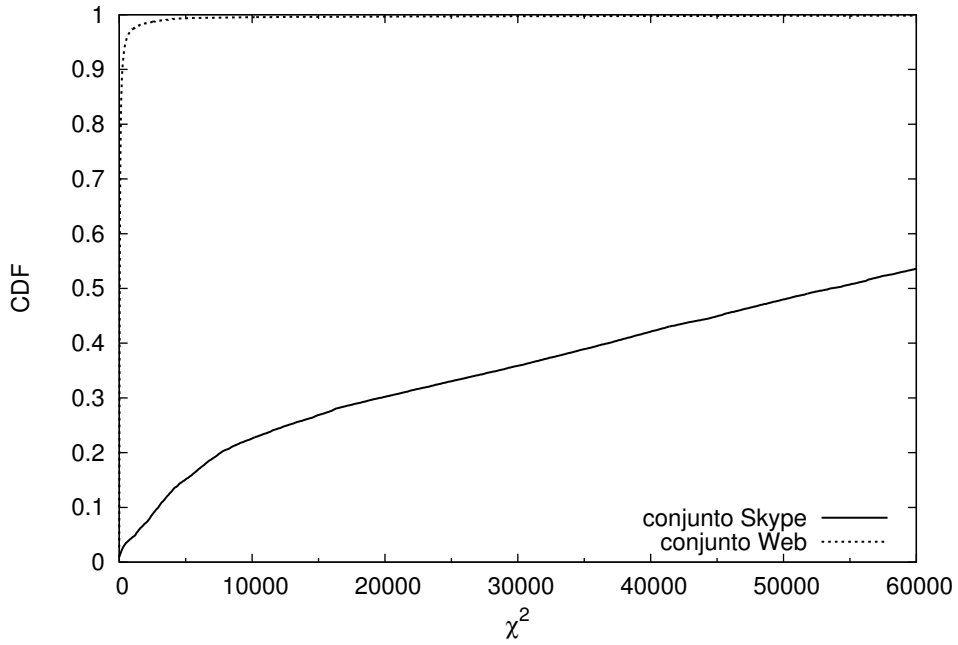

FIG. 4.7: CDF para χ^2 calculado a partir do intervalo entre requisições.

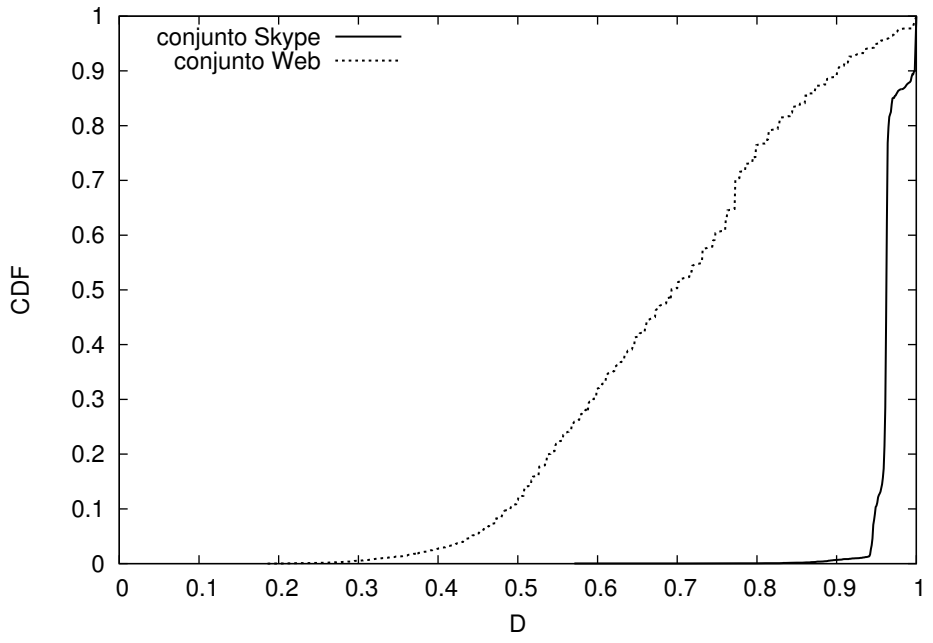

FIG. 4.8: CDF para D calculado a partir do tamanho da requisição.

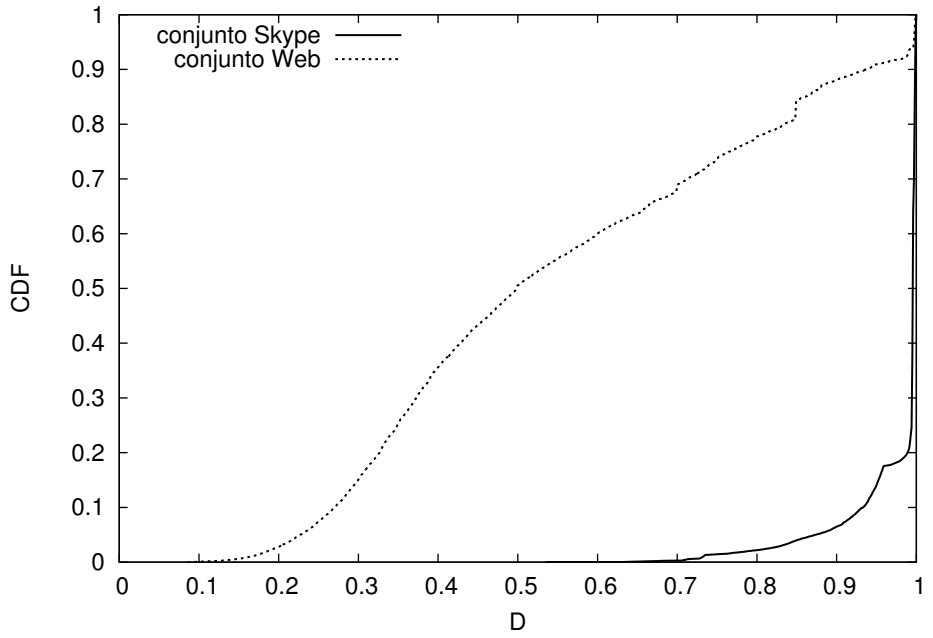

FIG. 4.9: CDF para D calculado a partir do tamanho da resposta.

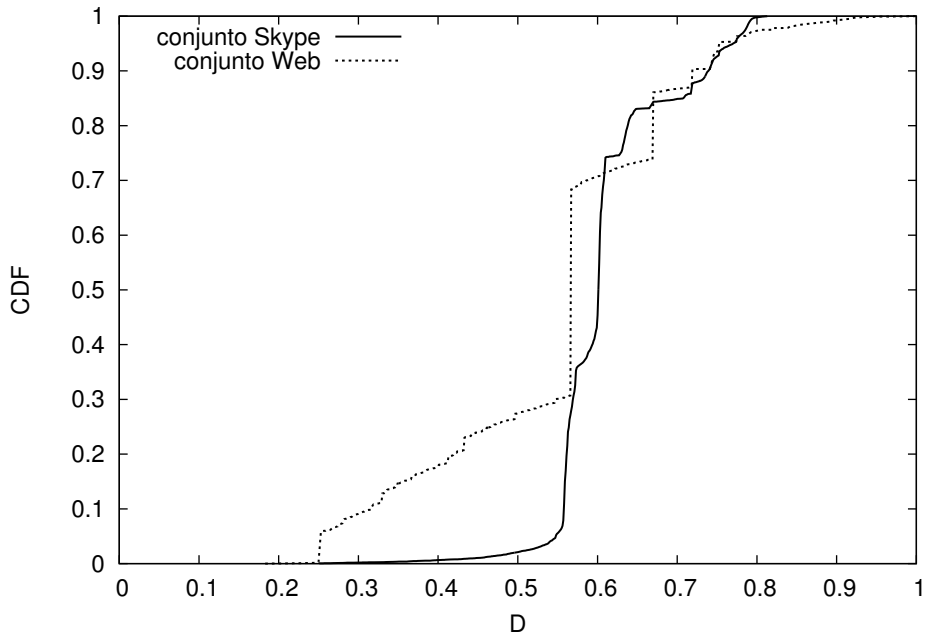

FIG. 4.10: CDF para D calculado a partir do intervalo entre requisições.

período de captura, ela iria trafegar com UDP e não seria capturada.

O tráfego Skype inserido para avaliação foi gerado por um pequeno conjunto de computadores que faziam parte da rede de clientes do provedor de acesso e possuíam *firewalls* individuais configurados para não permitir o uso do protocolo UDP para todas as portas, exceto DNS, nem o uso do TCP em portas diferentes de 80 ou 443. A partir dessas máquinas, foram feitas ligações para outras localizadas fora da referida rede. O programa Skype foi utilizado em versões da família 1.3 e 1.4 para Linux e 3.5 para Windows. As capturas ISP-D1 e ISP-D2 relacionadas na TAB. 4.4 seguiram essa metodologia.

Para a captura ISP-D3 foi utilizado o programa Google Talk em conjunto com o Skype para gerar chamadas VoIP. Novamente, supõe-se que não existe tráfego desses programas pelas portas Web além daquele produzido para avaliação, já que o Google Talk também só usa as portas Web na falta do protocolo UDP. O programa Google Talk não possui versão nativa ou equivalente para Linux que possibilite chamadas de voz, então todo o tráfego desse tipo foi gerado por máquinas com Windows.

Em nossos conjuntos de teste (ISP-D1, ISP-D2 e ISP-D3) foram capturados os cabeçalhos TCP/IP de todos os pacotes com porta origem ou destino igual a 80 ou 443. Informações sobre endereço IP, horário de início e duração foram registradas para todas as chamadas VoIP Skype ou Google Talk realizadas durante o período de captura. Após o término de cada captura, as informações foram usadas para uma identificação manual dos fluxos referentes as chamadas VoIP presentes no arquivo de captura. Essa identificação manual serviu como referência para avaliar a saída do nosso software.

4.3.1 COMBINAÇÃO DOS PARÂMETROS

Temos na nossa metodologia três parâmetros a serem avaliados: o tamanho das requisições, o tamanho das respostas e o intervalo de tempo entre requisições. Cada um deles pode ser avaliado separadamente e de forma independente dos outros, com a métrica χ^2 ou com a métrica D. Nesta seção estudamos se um resultado melhor é obtido com uma combinação desses parâmetros.

Foram realizados alguns testes com a captura ISP-D1 na busca de uma configuração que produzisse os melhores resultados. Primeiramente, cada um dos parâmetros foi ava-

liado individualmente e, em seguida, em combinações dois a dois. Também foi avaliada uma detecção que classificaria baseado no resultado da maioria, ou seja, na indicação de quaisquer dois dos três parâmetros. Por fim, foi testado o caso onde todos os parâmetros deveriam ter uma classificação positiva para o fluxo ser classificado positivamente.

Para a apresentação dos resultados, foram usadas curvas ROC. Cada ponto da curva é obtido com a execução do programa `testchi` ou `testks` com um conjunto diferente de limiares de detecção. A curva ROC fornece uma maneira de avaliar ao mesmo tempo os resultados de várias classificações, pois ela relaciona a sensibilidade e a taxa de falsos positivos de um classificador binário, onde os diversos pontos da curva são formados variando-se o limiar de detecção. A sensibilidade consiste no número de eventos positivos corretamente identificados como tal sobre o número total de eventos positivos possíveis de serem identificados, e a taxa de falsos positivos consiste no número de eventos negativos incorretamente identificados como tal sobre o número total de eventos negativos. Temos uma classificação mais precisa a medida que a curva se aproxima do canto superior esquerdo do gráfico. Uma classificação perfeita tem o maior valor possível para a sensibilidade (1) e o menor valor possível para a taxa de falsos positivos (0), ou seja, estaria localizada no ponto (0,1) do gráfico.

Antes de se determinar os pontos de cada curva, é necessário definir o conjunto de valores que será usado para a variação dos limiares de detecção, pois cada ponto da curva é obtido com um diferente limiar. No caso mais simples, isto é, das curvas que são geradas a partir de um único parâmetro, basta gerar uma lista ordenada de valores para servir como conjunto de limiares de decisão. Para a métrica χ^2, os valores da lista devem ser positivos, e para a métrica D, os valores da lista devem ser positivos e menores que 1. Com o conjunto de limiares definido, é realizada uma classificação binária baseado em cada valor e o procedimento é repetido diversas vezes, de acordo com o número de elementos do conjunto.

No caso onde existe mais de um parâmetro para ser avaliado, o procedimento de geração do conjunto de limiares é mais complexo, pois deve ser feita uma combinação entre as listas de valores de cada parâmetro para a geração do conjunto final. Vamos supor um caso onde três parâmetros são usados, e foram geradas três listas de valores

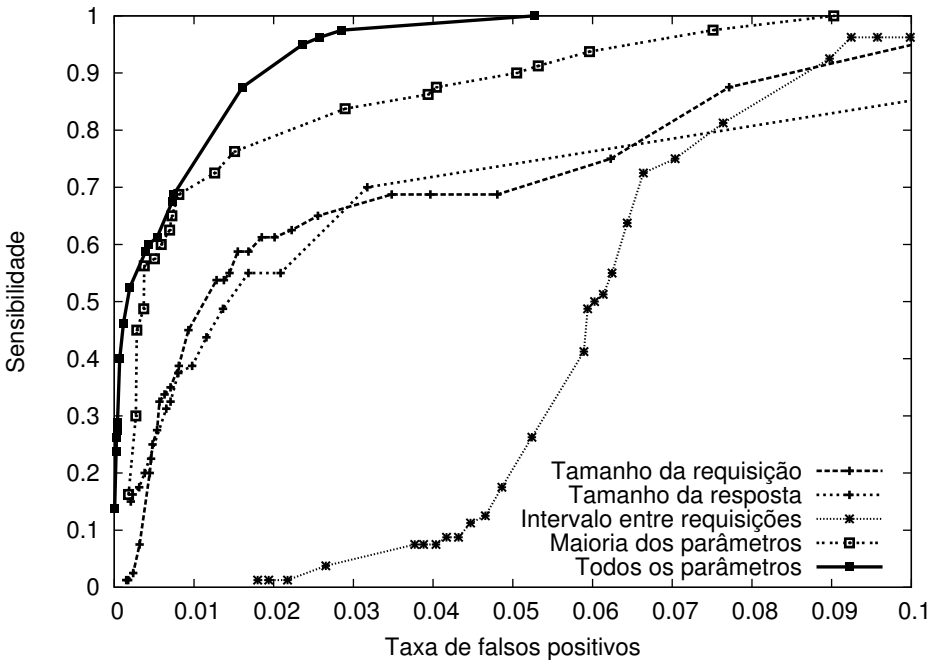

FIG. 4.11: Avaliação de diversas curvas ROC para detecção χ^2.

ordenados com número de elementos igual a n_1, n_2 e n_3, respectivamente. O número total de possíveis combinações que pode resultar dessas três listas é $N = n_1 \cdot n_2 \cdot n_3$.

A lista resultante com N elementos possui muitos elementos que, ao serem usados no programa, produzem uma taxa de falsos positivos bem superior em comparação com outros elementos, para uma mesma sensibilidade. Essa lista deve passar por refinamentos sucessivos até que os melhores elementos sejam selecionados e os outros sejam descartados. No conjunto final, devemos ter uma lista onde cada elemento possui três valores e cada conjunto de valores deve estar ordenado.

As curvas geradas para essa avaliação estão apresentadas na FIG. 4.11. A TAB. 4.5 mostra os valores obtidos para a captura ISP-D1 e detecção χ^2 com todos os parâmetros sendo necessários para uma classificação positiva. Cada linha representa um ponto da curva mostrada na FIG. 4.11 que foi gerada com todos os parâmetros.

Na TAB. 4.5, a primeira coluna indica a ordem do ponto gerado. As três colunas seguintes formam o conjunto de valores usado como limiar de detecção. Os limiares 1, 2, 3 representam os valores de comparação para a métrica χ^2 referente ao tamanho da requisição, tamanho da resposta e intervalo entre requisições, respectivamente. As

duas colunas finais apresentam a sensibilidade e a taxa de falsos positivos obtidos com a execução do programa, ou seja, as duas colunas finais representam as coordenadas de cada ponto no gráfico da FIG. 4.11.

TAB. 4.5: Pontos obtidos e limiares usados para a detecção χ^2 em ISP-D1.

Ponto	Limiar de detecção 1	Limiar de detecção 2	Limiar de detecção 3	Sensibilidade	Taxa de falsos positivos
1	3.000.000	100.000	5.000	0,1375	0,0001
2	3.000.000	50.000	5.000	0,2375	0,0002
3	2.000.000	50.000	5.000	0,2625	0,0003
4	2.000.000	50.000	2.000	0,2750	0,0003
5	1.500.000	50.000	2.000	0,2875	0,0004
6	1.500.000	25.000	1.000	0,4000	0,0007
7	1.500.000	10.000	1.000	0,4625	0,0011
8	1.000.000	10.000	1.000	0,5250	0,0019
9	500.000	8.000	800	0,5875	0,0039
10	500.000	5.000	750	0,6000	0,0043
11	250.000	5.000	750	0,6125	0,0054
12	100.000	5.000	750	0,6750	0,0073
13	100.000	4.000	500	0,6875	0,0075
14	20.000	1.000	500	0,8750	0,0161
15	10.000	1.000	500	0,9500	0,0236
16	10.000	1.000	250	0,9625	0,0258
17	10.000	500	200	0,9750	0,0285
18	5.000	200	100	1,0000	0,0527

Como pode ser visto na FIG. 4.11, cada parâmetro sozinho tem uma precisão de classificação menor em comparação com a solução conjunta. Para uma melhor visualização, nem todas as combinações testadas estão presentes na figura. Também foram testadas classificações baseadas em dois dos três parâmetros e uma classificação que necessitava de um resultado positivo nas duas métricas (χ^2 e D). Todos esses casos tiveram uma precisão inferior ao resultado obtido com os três parâmetros combinados. A partir desses resultados, o uso dos três parâmetros combinados foi adotado em todas as avaliações seguintes.

4.3.2 RESULTADOS

Após a escolha do método usado para a combinação dos parâmetros, foram gerados resultados de detecção para todas as capturas listadas na TAB. 4.4. Foram usadas as distribuições empíricas geradas na Seção 4.2 com cada arquivo capturado e o programa `dkfflow` para o cálculo dos valores χ^2 e D para cada fluxo. Após isso, foi usado o programa `testchi` na saída do programa `dkfflow`, juntamente com a lista de fluxos identificados manualmente como VoIP e o conjunto de limiares para detecção para a geração de um ponto da curva.

As curvas ROC para a detecção χ^2 são mostradas na FIG. 4.12. Os parâmetros foram calculados apenas para os fluxos com mais de 20 requisições, ou seja, o número de requisições por página foi usado como um filtro durante a execução do programa `dkfflow` para não levar em consideração os fluxos com poucas requisições. No total, foram selecionados 17.374 fluxos na captura ISP-D1 e 24.662 fluxos na captura ISP-D2 para o cálculo dos parâmetros.

A partir da FIG. 4.12 pode-se observar que todos os gráficos têm um comportamento similar, mas os resultados da captura ISP-D1 foram levemente melhores. A curva mostra que pode-se obter cerca de 90% dos 80 fluxos Skype identificados corretamente (sensibilidade de 0,9) com menos de 2% dos 17.294 fluxos não VoIP incorretamente identificados como tal (taxa de falsos positivos de 0,02). De maneira similar, também pode-se obter uma detecção de 100% dos fluxos VoIP com algo em torno de 5% de falsos positivos.

Uma simples mudança nos valores de decisão do programa `testchi` possibilita uma mudança na classificação. Caso se deseje um sistema de detecção mais conservador, com poucos falsos positivos encontrados (erros tipo I), deve-se escolher um ponto de trabalho próximo ao eixo Y, tendo como desvantagem uma menor taxa de acertos. Caso se deseje um sistema de detecção mais liberal, com uma taxa de acerto perto de 100%, ou seja, um sistema que produza um mínimo de falsos negativos (erros tipo II), deve-se escolher um ponto de trabalho próximo a linha de pontos com ordenada igual a 1, tendo como desvantagem um aumento da taxa de falsos positivos (erros tipo I).

Para a detecção Kolmogorov-Smirnov D, é realizado um procedimento similar, usando-se o programa `testks` no lugar do programa `testchi`. A execução do programa é feita

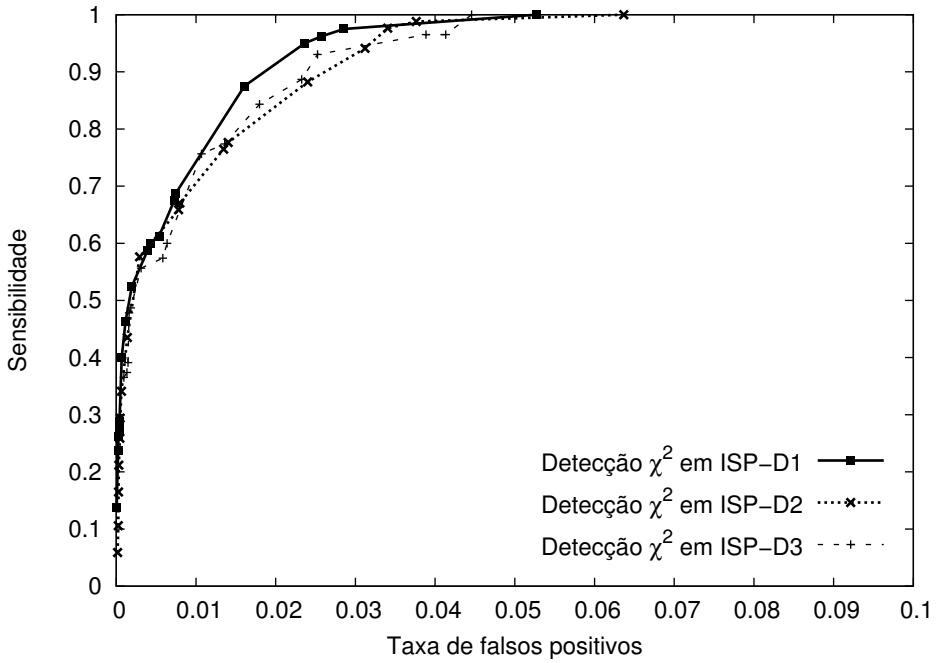

FIG. 4.12: Curvas ROC para detecção χ^2 em ISP-D1, ISP-D2 e ISP-D3.

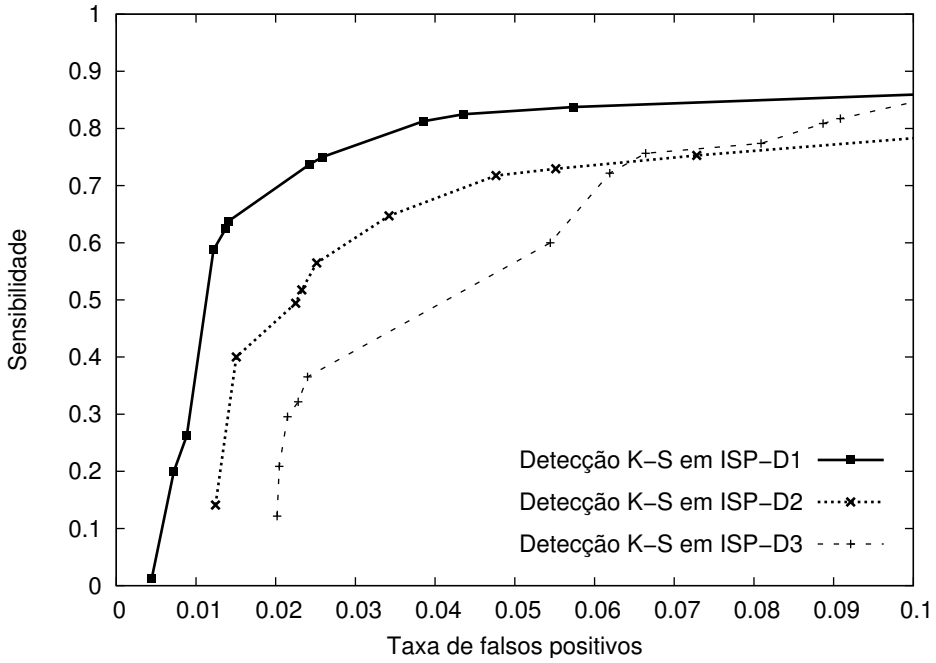

FIG. 4.13: Curvas ROC para detecção D em ISP-D1, ISP-D2 e ISP-D3.

de maneira equivalente, com a diferença que agora os limiares de detecção devem estar compreendidos no intervalo (0,1). Foram filtrados os fluxos com menos de 20 requisições e o número de fluxos selecionados para o cálculo das parâmetros foi o mesmo. Os resultados obtidos para as capturas listadas na TAB. 4.4 estão apresentados na FIG. 4.13.

Pela FIG. 4.13, observa-se que os melhores resultados foram obtidos com a captura ISP-D1. Com ela pode-se alcançar uma sensibilidade de 70% com uma taxa de falsos positivos em torno de 2% ou então 80% de sensibilidade com 5% de falsos positivos. As curvas relativas as outras capturas (ISP-D2 e ISP-D3) tiveram resultados menos expressivos.

A FIG. 4.12 e FIG. 4.13 encontram-se na mesma escala. Comparando-se seus resultados, observa-se que os resultados da detecção D não foram tão bons quanto aqueles obtidos pela detecção anterior pois os pontos da curva χ^2 apresentaram-se sempre mais próximos do canto superior esquerdo em comparação com as curva K-S. Esses resultados sugerem que a detecção χ^2 é possivelmente melhor do que a detecção K-S para tráfego VoIP usando a metodologia apresentada, algo que já havia sido indicado pelos resultados da Seção 4.2.

4.4 DETECÇÃO EM TEMPO REAL

Todos os resultados obtidos nas seções anteriores foram baseados em uma análise realizada muito tempo após a captura dos dados. Entretanto, um administrador de rede pode desejar identificar os fluxos VoIP que estão no momento usando os recursos da rede, não as ligações feitas alguns minutos ou horas atrás. Nessa seção, avalia-se a performance do algoritmo levando-se em consideração limitações de tempo.

Uma forma simples de levar o programa a dar respostas mais rápidas é limitar o tempo de captura. Desta forma, o sistema proposto para uma avaliação do algoritmo em uma situação de tempo real pode usar os mesmos dados colhidos na fase de treinamento, com uma diferente metodologia de detecção. Neste caso, cada análise do arquivo de captura deve ser feita de uma maneira relativamente rápida, de forma que uma classificação possa ser indicada antes que o referido fluxo entre em inatividade.

Escolheu-se inicialmente 10 segundos como um intervalo de tempo razoável entre o início da observação dos fluxos e a resposta do programa. Ligações VoIP normalmente

TAB. 4.6: Conjuntos gerados para simulação de uma situação de tempo real.

Nome	Origem	Número de arquivos	Duração de cada arquivo
RT1-10	ISP-D1 e ISP-D2	125	10s
RT1-30	ISP-D1 e ISP-D2	125	30s
RT1-60	ISP-D1 e ISP-D2	125	60s
RT2-10	ISP-D3	114	10s
RT2-30	ISP-D3	114	30s
RT2-60	ISP-D3	80	60s

são bem maiores que esse tempo e administradores de rede normalmente aceitariam uma espera de 10 segundos para receber uma classificação atualizada dos fluxos em um enlace. Assim, a cada 10 segundos o programa poderá analisar um novo conjunto de dados e atualizar suas informações de classificação dos fluxos IP.

Para um teste da viabilidade desse método, foram gerados novos conjuntos de dados. Diferentemente das vezes anteriores, esses novos conjunto de dados não foram capturados a partir do provedor de acesso mencionado anteriormente, mas foram dados extraídos das capturas ISP-D1, ISP-D2 e ISP-D3, conforme descrito na TAB. 4.6.

Para a geração do conjunto RT1-10, foram extraídos um total de 125 arquivos de captura, com 10 segundos cada, a partir das capturas ISP-D1 e ISP-D2. Esses 125 arquivos extraídos não foram de horários contíguos, mas sim separados por intervalos de tempo variados de forma a capturar diferentes ligações realizadas durante as capturas ISP-D1 e ISP-D2. Em uma situação de tempo real, as capturas poderiam ser seqüenciais ou até mesmo possuir algum intervalo de tempo comum, caso uma inicie antes do término da outra.

Após a geração de RT1-10, foram identificados manualmente todos os fluxos VoIP presentes para servir como referência para a detecção. Um total de 115 fluxos foram identificados e repetiu-se o procedimento descrito na Seção 4.3 para a geração dos pontos da curva ROC. A métrica D não foi utilizada por ter apresentado resultados inferiores ao sistema χ^2 na seção anterior.

Uma pequena mudança no programa `dkfflow` foi realizada para um melhor ajuste à nova situação. O valor mínimo para o número de requisições presentes em um fluxo foi

alterado para dez, devido ao pequeno tamanho dos arquivos de captura (10 segundos). O tempo necessário para a análise de cada arquivo de captura foi insignificante comparado a duração do arquivo (10s), utilizando-se o computador especificado no APÊNDICE 1 para os cálculos. Conforme descrito na Seção 3.4, os programas não foram desenvolvidos de uma forma otimizada. A curva ROC da detecção χ^2 em ISP-D1 também foi recalculada após essa mudança para comparação e os resultados dessa análise são mostrados na FIG. 4.14.

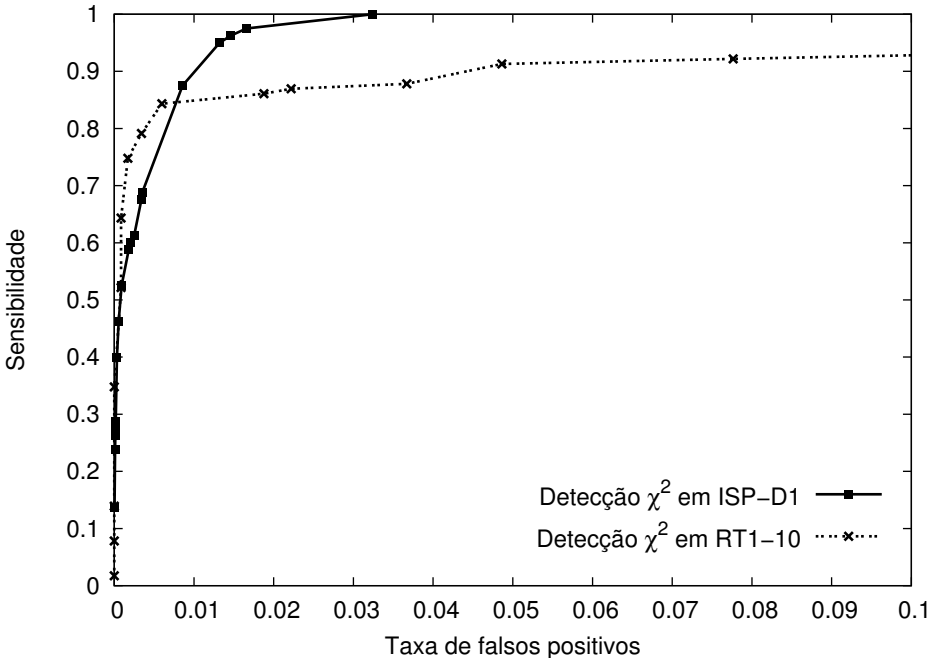

FIG. 4.14: Comparação entre detecção χ^2 em ISP-D1 e RT1-10.

Observa-se na FIG. 4.14 que a detecção χ^2 sobre o conjunto RT1-10 consegue uma sensibilidade de 85% com um número de falsos positivos comparável aos obtidos pela detecção χ^2 em ISP-D1. Além desse ponto, o número de falsos positivos aumenta consideravelmente sem um acréscimo significativo da sensibilidade, que se mantém em torno de 90%. Desta forma, a diferença entre as curvas é mais acentuada, com a combinação χ^2 ISP-D1 produzindo melhores resultados.

Esse resultado sugere que uma captura de 10 segundos já pode ser suficiente para uma detecção em tempo real, mas nessa situação deverão existir alguns fluxos que nunca serão identificados. Com um intervalo de captura maior, espera-se que a curva se aproxime mais da combinação χ^2 ISP-D1, pois o conjunto RT1-10 foi composto de amostras retiradas

das capturas ISP-D1 e ISP-D2.

Conforme apresentado na TAB. 4.6, foram gerados outros dois conjuntos de dados semelhantes a RT1-10, com a diferença na duração dos arquivos de captura. Em RT1-30 temos 125 arquivos de captura com 30 segundos cada e em RT1-60 outros 125 arquivos com 60 segundos cada. Seguindo-se o mesmo procedimento detalhado para RT1-10, foram gerados os pontos para uma detecção χ^2 em cada um deles.

Os resultados são mostrados na FIG. 4.15. Observa-se que com um intervalo de captura maior, aumenta-se a sensibilidade para uma mesma taxa de falsos positivos, resultado esse que já era esperado a partir da FIG. 4.14. Porém, mesmo a curva obtida a partir de RT1-60 ainda não consegue uma sensibilidade de 100% com uma taxa de falsos positivos em torno de 4%, obtida pela detecção em ISP-D1. A análise de conjuntos com intervalos maiores possivelmente resultaria em intervalos mais próximos aos obtidos por ISP-D1, mas já não seria mais uma classificação que atendesse os objetivos de tempo dessa seção.

Também foram gerados três outros conjuntos de dados para a avaliação da detecção proposta e os resultados são apresentados na FIG. 4.16. Como descrito na TAB. 4.6, RT2-10, RT2-30 e RT2-60, foram criados a partir de ISP-D3, a captura que reuniu chamadas VoIP feitas pelo Skype e pelo Google Talk, com arquivos de 10s, 30s e 60s, respectivamente. Neste caso foi novamente utlizada apenas a detecção χ^2 .

Pela figura, observa-se que não existe uma separação muito clara entre os resultados obtidos em RT2-30 e RT2-60, mas a detecção em RT2-10 foi levemente inferior. Com intervalos de 10 segundos, pode-se conseguir uma sensibilidade de 90% com uma taxa de falsos positivos em torno de 2,5%. Uma sensibilidade próxima a 100% já pode ser alcançada com uma taxa de falsos positivos de 5% ou mais.

Os resultados dessa seção sintetizados pela FIG. 4.15 e FIG. 4.16 sugerem que a proposta apresentada para uma detecção em tempo real é factível e pode atingir resultados comparáveis à metodologia original.

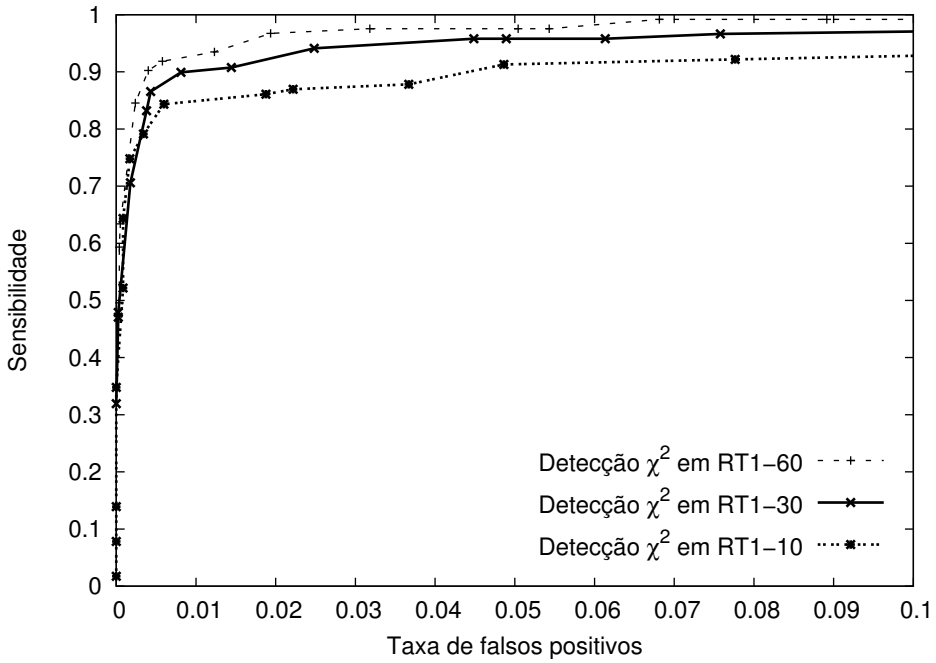

FIG. 4.15: Curvas ROC para detecção χ^2 em RT1-10, RT1-30 e RT1-60.

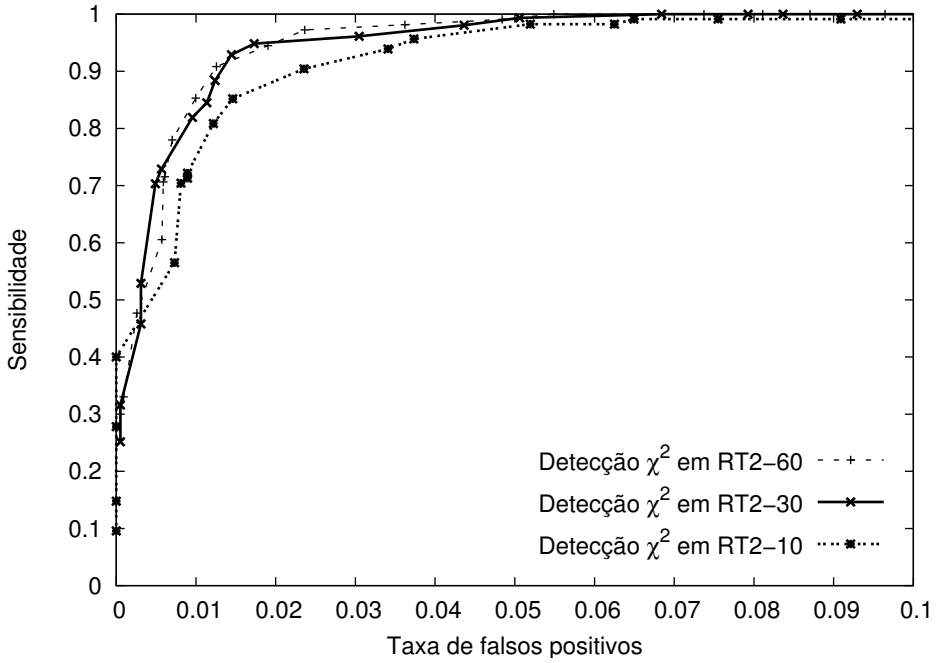

FIG. 4.16: Curvas ROC para detecção χ^2 em RT2-10, RT2-30 e RT2-60.

5 CONCLUSÃO

Nesse trabalho foi apresentada uma metodologia para a detecção de fluxos anômalos em tráfego HTTP. Consideramos que a utilidade do protocolo HTTP é o acesso de páginas WWW. Desta forma, podem ser consideradas como anomalias o trânsito de algum outro protocolo de aplicação pelas portas HTTP (80 e 443/TCP) ou o uso do HTTP com finalidades diferentes do acesso de páginas WWW. Durante o ano de 2007, foram produzidos a partir deste trabalho dois artigos científicos aceitos para publicação em congressos internacionais (FREIRE, 2007) e (FREIRE, 2008).

5.1 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentada uma metodologia para detecção de anomalias em amostras de fluxos IP do protocolo HTTP/HTTPS, que é responsável por uma fração significativa do tráfego total da Internet. Para tanto, foi desenvolvida uma metodologia de detecção baseada em um modelo de tráfego HTTP. A metodologia desenvolvida foi avaliada para a detecção de fluxos VoIP em HTTP, com o uso dos programas Skype e Google Talk para geração de tráfego VoIP. Embora a metodologia tenha sido avaliada apenas para fluxos VoIP, ela não leva em consideração informações específicas de algum programa nem é baseada na análise do conteúdo dos pacotes a serem classificados. Devido a grande popularidade de programas VoIP atualmente, optou-se por usar fluxos VoIP como um alvo primário para o teste da metodologia. Como trabalho futuro, pode-se pensar no mesmo método sendo usado para o reconhecimento de outros tipos de fluxos anômalos.

Um modelo de tráfego HTTP foi desenvolvido levando-se em consideração modelos desenvolvidos em trabalhos anteriores e com parâmetros relativamente simples de serem calculados. Nosso modelo foi povoado com dados reais e conta com cinco parâmetros. Três parâmetros (tamanho da requisição, tamanho da resposta e intervalo entre requisições) foram usados na geração de distribuições empíricas para uma caracterização do tráfego HTTP considerado normal e outros dois parâmetros (número de requisições por página

e tempo de carregamento da página) foram usados em operações auxiliares. No caso do uso do HTTP para transferência de grandes arquivos, da ordem de dezenas ou centenas de Megabytes, não foram realizados experimentos para avaliar a detecção desse tipo de tráfego mas sua identificação seria trivial, uma vez que o tamanho da resposta é um parâmetro do modelo HTTP utilizado.

Na captura e tratamento dos dados foram usados programas com licença GPL disponíveis na Internet (`tcpdump`, `tcpflow`) e foram desenvolvidas novas ferramentas para tarefas específicas desse trabalho (`ckfflow`, `dkfflow`, `testchi`). Como trabalho futuro, essas ferramentas desenvolvidas podem ser melhoradas e otimizadas para que possam ser disponibilizadas para o público.

Para a comparação dos fluxos IP com o modelo de tráfego HTTP, foram usadas duas métricas retiradas de dois testes estatísticos: o valor χ^2 do teste Chi-quadrado e a distância D do teste Kolmogorov-Smirnov. Foram usadas curvas ROC para apresentação dos resultados, mostrando que os programas desenvolvidos podem ser ajustados para uma detecção mais rigorosa ou não. Em outras palavras, um administrador de rede pode configurar o programa para uma maior ou menor sensibilidade de detecção, obtendo por conseqüência uma maior ou menor taxa de falsos positivos.

Na avaliação dos resultados, foram colhidos dados reais de um provedor de acesso Internet e uma instituição acadêmica em diferentes períodos de tempo. Esses conjuntos de dados apresentaram um comportamento semelhante, sugerindo que os dados de treinamento colhidos permaneceram válidos durante todo o período de avaliação, que durou alguns meses. As distribuições empíricas geradas do provedor de acesso foram bem semelhantes entre si, e a distribuição gerada a partir da instituição acadêmica apresentou pequenas diferenças, indicando uma leve diferença no comportamento médio do tráfego. No provedor de acesso, cada usuário está sujeito a uma limitação de banda e no ambiente acadêmico não existe essa limitação para cada usuário individual, ou seja, transferências TCP nesse ambiente podem alcançar taxas de transmissão bem maiores.

Os resultados experimentais mostraram uma boa eficiência do algoritmo para detecção VoIP, obtendo uma detecção de 90% dos fluxos possíveis de serem identificados com algo em torno de 2% de fluxos HTTP não identificados como tal. Dependendo da situação,

uma taxa de detecção em torno de 100% pode ser alcançada como cerca de 5% de falsos positivos. Também foi avaliada essa metodologia em uma simulação de detecção com restrições de tempo, e os resultados obtidos foram comparáveis com os da detecção sem limitação de tempo.

A análise "post-mortem" dos dados seria interessante para objetivos como a formação de dados estatísticos ou a identificação dos principais geradores desse tipo de tráfego. A identificação em tempo real tem como possibilidade adicional a escolha de uma ação imediata para ser tomada após a identificação de um fluxo, como por exemplo, bloquear todo tráfego identificado como VoIP ou dar a esse tipo de tráfego um enlace prioritário. Nesse último caso, os resultados sugerem que já poderiam ser apresentadas respostas cerca de 10 segundos após o ínicio da observação do tráfego.

5.2 TRABALHOS FUTUROS

O presente trabalho não deu continuidade a trabalhos anteriores mas os estudos e resultados obtidos aqui podem ser aprimorados em trabalhos futuros. Existem diversos tipos possíveis de anomalias de redes, neste trabalho atacou-se o caso específico de outros protocolos em tráfego HTTP. Uma primeira sugestão para trabalhos futuros seria a avaliação do método na detecção de outros tipos de fluxos anômalos. Uma vez que possam ser detectados diversos tipos de anomalias com a mesma metodologia, deve-se pensar em como classificar esses diversos tipos, ou seja, realizar uma classificação com o resultado da detecção.

O modelo HTTP definido também pode ser expandido, de forma a guardar outros detalhes do protocolo HTTP que poderiam auxiliar a detecção. Outra sugestão seria verificar se o comportamento HTTP observado de cada um dos parâmetros representa um exemplo típico de tráfego HTTP ou se dados colhidos de outras fontes teriam um características bem diferentes. Além disso, pode-se investigar a validade das distribuições empíricas ou se os dados colhidos em uma determinada rede podem ser usados para detecção de anomalias em outra rede.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHAMSSON, Henrik e AHLGREN, Bengt. **Using empirical distributions to characterize web client traffic and to generate synthetic traffic**. Em GLOBECOM'00: Proceedings of IEEE Global Telecommunications Conference, v. 1, p. 428–433, 2000. ISBN 0-7803-6451-1.

ALARCON-AQUINO, Vicente e BARRIA, Javier A. **Anomaly detection in communication networks using wavelets**. IEE Proceedings Communications, v. 148, n. 6, p. 355–362, 2001. ISSN 1350-2425.

ARLITT, Martin e WILLIAMSON, Carey. **A synthetic workload model for internet mosaic traffic**. Em Proceedings of the 1995 Summer Computer Simulation Conference, p. 852–857, 1995.

BARFORD, Paul e CROVELLA, Mark. **Generating representative web workloads for network and server performance evaluation**. Em SIGMETRICS'98: Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, p. 151–160, 1998. ISBN 0-89791-982-3.

BARFORD, Paul, KLINE, Jeffery, PLONKA, David e RON, Amos. **A signal analysis of network traffic anomalies**. Em IMW'02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, p. 71–82, 2002. ISBN 1-58113-603-X.

BARFORD, Paul e PLONKA, David. **Characteristics of network traffic flow anomalies**. Em IMW'01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, p. 69–73, 2001. ISBN 1-58113-435-5.

BASET, Salman A. e SCHULZRINNE, Henning G. **An analysis of the skype peer-to-peer internet telephony protocol**. Em INFOCOM'06: Proceedings of the 25th IEEE International Conference on Computer Communications, p. 1–11, 2006. ISBN 1-4244-0221-2.

BERNAILLE, Laurent, TEIXEIRA, Renata, AKODKENOU, Ismael, SOULE, Augustin e SALAMATIAN, Kavé. **Traffic classification on the fly**. SIGCOMM Comput. Commun. Rev., v. 36, n. 2, p. 23–26, 2006. ISSN 0146-4833.

BERNERS-LEE, Tim, FIELDING, Roy e FRYSTYK, Henrik. **Hypertext transfer protocol - HTTP/1.0**. Internet Request for Comments 1945, 1996.

BONFIGLIO, Dario, MELLIA, Marco, MEO, Michela, ROSSI, Dario e TOFANELLI, Paolo. **Revealing skype traffic: when randomness plays with you**. SIGCOMM Comput. Commun. Rev., v. 37, n. 4, p. 37–48, 2007. ISSN 0146-4833.

CAO, Jin, CLEVELAND, William S., LIN, Dong e SUN, Don X. **Internet traffic tends toward Poisson and independent as the load increases**. Nonlinear Estimation and Classification, p. 83–109, 2002. D. Denison, M. Hansen, C. Holmes, B. Mallick, and B. Yu, Eds.

CHOI, Hyoung-Kee e LIMB, John O. **A behavioral model of web traffic**. Em ICNP'99: Proceedings of the 7th International Conference on Network Protocols, p. 327–334. IEEE Computer Society, 1999. ISBN 0-7695-0412-4.

COCHRAN, William G. **The chi-square test of goodness of fit**. Annals of Mathematical Statistics, v. 23, p. 315–345, 1952.

CROVELLA, Mark E. e BESTAVROS, Azer. **Self-similarity in World Wide Web traffic: evidence and possible causes**. IEEE /ACM Transactions on Networking, v. 5, n. 6, p. 835–846, 1997. ISSN 1063-6692.

DASGUPTA, Dipankar e GONZÁLEZ, Fabio. **An immunity-based technique to characterize intrusions in computer networks**. IEEE Transactions on Evolutionary Computation, v. 6, n. 3, p. 281–291, 2002.

EHLERT, Sven, PETGANG, Sandrine, MAGEDANZ, Thomas e SISALEM, Dorgham. **Analysis and signature of skype VoIP session traffic**. Em CIIT 2006: 4th IASTED International Conference on Communications, Internet, and Information Technology, p. 83–89, 2006. ISBN 0-88986-613-9.

ESTÉVEZ-TAPIADOR, Juan M., GARCÍA-TEODORO, Pedro e DÍAZ-VERDEJO, Jesús E. **Measuring normality in http traffic for anomaly-based intrusion detection**. Computer Networks, v. 45, n. 2, p. 175–193, 2004. ISSN 1389-1286.

FIELDING, Roy, GETTYS, James, MOGUL, Jeffrey, FRYSTYK, Henrik, MASINTER, Larry, LEACH, Paul e BERNERS-LEE, Tim. **Hypertext transfer protocol - HTTP/1.1**. Internet Request for Comments 2616, 1999.

FRALEIGH, Chuck, MOON, Sue, LYLES, Bryan, COTTON, Chase, KHAN, Mujahid, MOLL, Deb, ROCKELL, Rob, SEELY, Ted e DIOT, Christophe. **Packet-level traffic measurements from the sprint ip backbone**. IEEE Network, v. 17, n. 6, p. 6–16, 2003. ISSN 0890-8044.

FREIRE, Emanuel P., ZIVIANI, Artur e SALLES, Ronaldo M. **On metrics to distinguish skype flows from HTTP traffic**. Em LANOMS 2007: Proceedings of the 5th Latin American Network Operations and Management Symposium, p. 57–66, 2007. ISBN 978-1-4244-1182-5.

FREIRE, Emanuel P., ZIVIANI, Artur e SALLES, Ronaldo M. **Detecting skype flows in web traffic**. Em NOMS 2008: Proceedings of the 2008 IEEE/IFIP Network Operations and Management Symposium, 2008.

GUHA, Saikat, DASWANI, Neil e JAIN, Ravi. **An experimental study of the skype peer-to-peer VoIP system**. Em IPTPS'06: Proceedings of the 5th International Workshop on Peer-to-Peer Systems, p. 1–6, 2006.

HANDLEY, Ed., Mark, RESCORLA, Ed., Eric e IAB. **Internet denial-of-service considerations**. Internet Request for Comments 4732, 2006.

HARMER, Paul K., WILLIAMS, Paul D., GUNSCH, Gregg H. e LAMONT, Gary B. **An artificial immune system architecture for computer security applications**. IEEE Transactions on Evolutionary Computation, v. 6, n. 3, p. 252–280, 2002.

HUSSAIN, Alefiya, HEIDEMANN, John e PAPADOPOULOS, Christos. **A framework for classifying denial of service attacks**. Em SIGCOMM'03: Proceedings of the 2003 ACM conference on Applications, technologies, architectures, and protocols for computer communications, p. 99–110, 2003. ISBN 1-58113-735-4.

JUNG, Jaeyeon, KRISHNAMURTHY, Balachander e RABINOVICH, Michael. **Flash crowds and denial of service attacks: characterization and implications for cdns and web sites**. Em WWW'02: Proceedings of the 11th international conference on World Wide Web, p. 293–304, 2002. ISBN 1-58113-449-5.

KARAGIANNIS, Thomas, BROIDO, Andre, BROWNLEE, Nevil, CLAFFY, Kimberly C. e FALOUTSOS, Michalis. **Is P2P dying or just hiding?** Em GLOBECOM 2004: Proceedings of IEEE Global Telecommunications Conference, 2004a.

KARAGIANNIS, Thomas, MOLLE, Mart, FALOUTSOS, Michalis e BROIDO, Andre. **A Nonstationary Poisson View of Internet Traffic**. Em INFOCOM'04: Proceedings of the 23rd IEEE International Conference on Computer Communications, 2004b.

KARAGIANNIS, Thomas, PAPAGIANNAKI, Konstantina e FALOUTSOS, Michalis. **Blinc: multilevel traffic classification in the dark**. Em SIGCOMM'05: Proceedings of the 2005 ACM conference on Applications, technologies, architectures, and protocols for computer communications, p. 229–240, 2005. ISBN 1-59593-009-4.

KRISHNAMURPHY, Balachander, MOGUL, Jeffrey C. e KRISTOL, David M. **Key differences between HTTP/1.0 and HTTP/1.1**. Em WWW'99: Proceeding of the eighth international conference on World Wide Web, p. 1737–1751. Elsevier North-Holland, Inc., 1999.

KRUGEL, Christopher, TOTH, Thomas e KIRDA, Engin. **Service specific anomaly detection for network intrusion detection**. Em SAC'02: Proceedings of the 2002 ACM symposium on Applied computing, p. 201–208, 2002. ISBN 1-58113-445-2.

LAKHINA, Anukool, CROVELLA, Mark e DIOT, Christiphe. **Characterization of network-wide anomalies in traffic flows**. Em IMC'04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, p. 201–206, 2004a. ISBN 1-58113-821-0.

LAKHINA, Anukool, CROVELLA, Mark e DIOT, Christophe. **Diagnosing network-wide traffic anomalies**. Em SIGCOMM'04: Proceedings of the 2004 ACM conference on Applications, technologies, architectures, and protocols for computer communications, p. 219–230, 2004b. ISBN 1-58113-862-8.

LAKHINA, Anukool, CROVELLA, Mark e DIOT, Christophe. **Mining anomalies using traffic feature distributions**. Em SIGCOMM'05: Proceedings of the 2005 ACM conference on Applications, technologies, architectures, and protocols for computer communications, p. 217–228, 2005. ISBN 1-59593-009-4.

LEE, Wenke e XIANG, Dong. **Information-theoretic measures for anomaly detection**. Em SP'01: Proceedings of the 2001 IEEE Symposium on Security and Privacy, p. 130–143, 2001.

LELAND, Will E., TAQQU, Murad S., WILLINGER, Walter e WILSON, Daniel V. **On the self-similar nature of ethernet traffic (extended version)**. IEEE /ACM Transactions on Networking, v. 2, n. 1, p. 1–15, 1994.

LI, Xin, BIAN, Fang, CROVELLA, Mark, DIOT, Christophe, GOVINDAN, Ramesh, IANNACCONE, Gianluca e LAKHINA, Anukool. **Detection and identification of network anomalies using sketch subspaces**. Em IMC'06: Proceedings of the 6th ACM SIGCOMM on Internet measurement, p. 147–152, 2006. ISBN 1-59593-561-4.

MA, Justin, LEVCHENKO, Kirill, KREIBICH, Christian, SAVAGE, Stefan e VOELKER, Geoffrey M. **Unexpected means of protocol inference**. Em IMC'06: Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, p. 313–326, 2006. ISBN 1-59593-561-4.

MAH, Bruce A. **An empirical model of HTTP network traffic**. Em INFOCOM'97: Proceedings of 16th Joint Conference of the IEEE Computer and Communications Societies, 1997. ISBN 0-8186-7780-5.

MASSEY, Jr., Frank J. **The Kolmogorov-Smirnov test of goodness of fit**. Journal of the American Statistical Association, v. 46, p. 68–78, 1951.

MOORE, Andrew e PAPAGIANNAKI, Konstantina. **Toward the accurate identification of network applications**. Em PAM2005: Proceedings of the Passive and Active Measurement Workshop, 2005. ISBN 3-540-25520-6.

MOORE, David, VOELKER, Geoffrey M. e SAVAGE, Stefan. **Inferring internet Denial-of-Service activity**. Em Proceedings of the 10th USENIX Security Symposium, p. 9–22, 2001.

NIVEN, Larry. **Flight of the Horse, The**, chapter Flash Crowd. Ballantine Books, 1973.

PAXSON, Vern. **Bro: a system for detecting network intruders in real-time**. Computer Networks (Amsterdam, Netherlands: 1999), v. 31, n. 23–24, p. 2435–2463, 1999.

PAXSON, Vern e FLOYD, Sally. **Wide area traffic: the failure of Poisson modeling**. IEEE/ACM Transactions on Networking, v. 3, n. 3, p. 226–244, 1995.

PAXSON, Vern e FLOYD, Sally. **Why we don't know how to simulate the internet**. Em WSC'97: Proceedings of the 29th conference on Winter simulation, p. 1037–1044, 1997. ISBN 0-7803-4278-X.

PLONKA, Dave. **Flowscan: A network traffic flow reporting and visualization tool**. Em LISA '00: Proceedings of the 14th USENIX conference on System administration, p. 305–318, 2000.

SEREDYNSKI, Franciszek e BOUVRY, Pascal. **Anomaly detection in TCP/IP networks using the immune systems paradigm**. Computer Communications, v. 30, n. 4, p. 740–749, 2007.

SOULE, Augustin, SALAMATIAN, Kavé e TAFT, Nina. **Combining filtering and statistical methods for anomaly detection**. Em IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet measurement, p. 1–14, 2005.

SRIPANIDKULCHAI, Kunwadee, MAGGS, Bruce e ZHANG, Hui. **An analysis of live streaming workloads on the internet**. Em IMC'04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, p. 41–54, 2004. ISBN 1-58113-821-0.

SUH, Kyoungwon, FIGUEIREDO, Daniel R., KUROSE, Jim e TOWSLEY, Don. **Characterizing and detecting relayed traffic: A case study using skype**. Em INFOCOM'06: Proceedings of the 25th IEEE International Conference on Computer Communications, 2006.

WILLINGER, Walter, PAXSON, Vern e TAQQU, Murad S. **Self-similarity and Heavy Tails: Structural Modeling of Network Traffic**. A Practical Guide to Heavy Tails: Statistical Techniques and Applications, 1998. Adler, R., Feldman, R., and Taqqu, M.S., Eds.

XU, Kuai, ZHANG, Zhi-Li e BHATTACHARYYA, Supratik. **Profiling internet backbone traffic: behavior models and applications**. Em SIGCOMM'05: Proceedings of the 2005 ACM conference on Applications, technologies, architectures, and protocols for computer communications, p. 169–180, 2005. ISBN 1-59593-009-4.

YE, Nong e CHEN, Qiang. **An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems**. Quality and Reliability Engineering International, v. 17, n. 2, p. 105–112, 2001a.

YE, Nong, LI, Xiangyang, CHEN, Qiang, EMRAN, Syed Masum e XU, Mingming. **Probabilistic techniques for intrusion detection based on computer audit data**. IEEE Transactions on Systems, Man and Cybernetics, Part A, v. 31, n. 4, p. 266–274, 2001b.

ZHANG, Yin, GE, Zihui, GREENBERG, Albert e ROUGHAN, Matthew. **Network anomography**. Em IMC'05: Proceedings of the 5th ACM SIGCOMM conference on Internet measurement, p. 1–14, 2005.

7 APÊNDICES

7.1 APÊNDICE 1: PROCEDIMENTOS DE CAPTURA DE PACOTES

- Computador utilizado para captura no provedor de acesso

Processador AMD Sempron 64 2800

Memória RAM de 512 MB

2 discos rígidos IDE de 250 GB

1 placa de rede PCI fast Ethernet

1 placa de rede gigabit Ethernet integrada à placa mãe

Sistema operacional Fedora Core 6 64 bits

Uma das placas de rede é ligada ao switch configurado para realizar o espelhamento da porta do enlace Internet e a outra é utilizada para acesso remoto ao computador.

- Geração das distribuições empíricas

a) Captura dos pacotes completos para os dados de treinamento:

```
# nohup tcpdump port 80 -i eth1 -C 2000 -W 100 -s 0 -w arquivo &
```

Com o programa `tcpdump`, é feita uma captura dos pacotes TCP com porta de origem ou destino igual a 80. É realizada uma captura do conteúdo completo dos pacotes TCP (`-s 0`).

b) Cálculo dos parâmetros para cada arquivo de captura:

```
# ckfflow -r arquivo00 req.txt rsp.txt itv.txt nrq.txt pgt.txt
# ckfflow -r arquivo01 req.txt rsp.txt itv.txt nrq.txt pgt.txt
...
# ckfflow -r arquivo99 req.txt rsp.txt itv.txt nrq.txt pgt.txt
```

O programa `ckfflow` lê o arquivo de captura gerado pelo `tcpdump` e produz cinco arquivos texto como saída (`req.txt`, `rsp.txt`, `itv.txt`, `nrq.txt` e `pgt.txt`). Cada arquivo é relativo a um parâmetro do modelo e possui um valor por linha. Para a geração de uma distribuição é necessário ordenar os elementos do arquivo, definir intervalos de valores para os elementos do arquivo e contar o número de elementos em cada intervalo considerado.

- Detecção de anomalias

a) Captura dos cabeçalhos dos pacotes:

```
# nohup tcpdump port 80 or port 443 -i eth1 -C 2000 -W 100 -w arq &
```

Com o programa `tcpdump`, é feita uma captura dos pacotes TCP com porta de origem ou destino igual a 80 ou 443. É realizada uma captura no tamanho padrão do `tcpdump` (68 bytes de cada pacote).

b) Cálculo dos parâmetros para cada arquivo de captura:

```
# dkfflow -r arq00 -d1 d1.txt -d2 d2.txt -d3 d3.txt out.txt flw.txt

# dkfflow -r arq01 -d1 d1.txt -d2 d2.txt -d3 d3.txt out.txt flw.txt

...

# dkfflow -r arq99 -d1 d1.txt -d2 d2.txt -d3 d3.txt out.txt flw.txt
```

Os arquivos `d1.txt`, `d2.txt` e `d3.txt` representam as distribuições geradas anteriormente para o tamanho das requisições, tamanho das respostas e intervalo entre requisições, respectivamente. Os dois últimos arquivos (`out.txt` e `flw.txt`) são arquivos texto de saída, o primeiro contém os parâmetros χ^2 e D calculados e o segundo apresenta informações sobre todos os fluxos identificados (IP origem e destino, porta origem e destino, hora de início e duração).

c) Identificação dos fluxos anômalos:

O programa `testchi` (ou `testks`) deve ser executado com o arquivo texto que contém os valores χ^2 e D calculados (`out.txt`) e o arquivo com os valores dos limiares de detecção (`valores.txt`).

```
# testchi -r out.txt -t valores.txt saida-p.txt saida-n.txt
```

Os arquivos `saida-p.txt` e `saida-n.txt` são arquivos texto de saída, o primeiro contém informações sobre os fluxos identificados como anômalos e o segundo informações sobre os fluxos identificados como normais. A partir do arquivo `flw.txt` definido anteriormente, é feita um identificação manual dos fluxos anômalos e os

resultados são comparados com os arquivos `saida-p.txt` e `saida-n.txt` para a determinação da taxa de acerto e taxa de falsos positivos.

7.2 APÊNDICE 2: DADOS DAS DISTRIBUIÇÕES EMPÍRICAS

TAB. 7.1: Valores X e Y das curvas presentes na FIG. 4.2.

X	Y ISP-T1	Y ISP-T2	Y ISP-T3	Y ACD-T1
20	0,003446	0,001319	0,000000	0,000002
50	0,003544	0,001413	0,000056	0,000020
80	0,003614	0,002474	0,000205	0,000141
110	0,004722	0,003471	0,002127	0,000786
140	0,006474	0,005309	0,004806	0,002429
170	0,008105	0,007268	0,006855	0,007490
200	0,010714	0,011930	0,013275	0,011240
230	0,015899	0,017934	0,021960	0,022822
260	0,028273	0,030668	0,031071	0,032951
290	0,063186	0,130274	0,065120	0,052696
320	0,113654	0,181705	0,103567	0,079384
350	0,163427	0,227386	0,148427	0,114578
380	0,215912	0,272892	0,191984	0,148907
410	0,262867	0,316432	0,238851	0,187919
440	0,315772	0,360859	0,289355	0,245586
470	0,371273	0,410575	0,341662	0,297431
500	0,460881	0,485528	0,418874	0,354150
530	0,527271	0,549640	0,482436	0,401999
560	0,578402	0,599945	0,538506	0,449070
590	0,621105	0,643372	0,589100	0,492586
620	0,655612	0,676341	0,637291	0,536404
650	0,692423	0,707790	0,677883	0,577782
680	0,719326	0,735562	0,705924	0,624267
710	0,742852	0,756766	0,729055	0,662680
740	0,760884	0,774136	0,749695	0,693527
770	0,776420	0,789774	0,768497	0,714308
800	0,789619	0,805236	0,785270	0,737020
830	0,801207	0,816707	0,798181	0,751529
860	0,812126	0,826548	0,811696	0,762980
890	0,821826	0,836686	0,823065	0,779473
920	0,832915	0,845926	0,833827	0,791095
950	0,843798	0,855606	0,845484	0,804362
980	0,854809	0,866526	0,857685	0,815150
1010	0,865212	0,876590	0,869461	0,839506
1040	0,875029	0,886535	0,880657	0,850546
1070	0,885309	0,896237	0,891345	0,860897
1100	0,895473	0,905755	0,902740	0,873620
1130	0,906556	0,916727	0,913752	0,882976
1160	0,918205	0,927607	0,925153	0,890816
1190	0,930174	0,939564	0,935693	0,899946
1220	0,939282	0,947978	0,944819	0,908647
1250	0,946451	0,954252	0,952321	0,918826
1280	0,952031	0,959646	0,958182	0,926703
1310	0,956882	0,964387	0,962960	0,932851
1340	0,961709	0,968371	0,967022	0,937782
1370	0,965338	0,972176	0,970453	0,943342
1400	0,970745	0,975715	0,973818	0,950131

TAB. 7.2: Valores X e Y das curvas presentes na FIG. 4.3.

X	Y ISP-T1	Y ISP-T2	Y ISP-T3	Y ACD-T1
100	0,000001	0,000016	0,000018	0,000009
200	0,003915	0,005669	0,004740	0,005298
300	0,034159	0,033327	0,033661	0,063931
400	0,104655	0,105346	0,104711	0,137098
500	0,156739	0,160112	0,151270	0,188186
600	0,197118	0,200921	0,188506	0,251243
700	0,228562	0,233686	0,213941	0,281929
800	0,255105	0,260348	0,235191	0,305563
900	0,283417	0,289058	0,262805	0,326893
1000	0,304673	0,308315	0,278984	0,346619
1100	0,321496	0,325675	0,294837	0,365511
1200	0,339750	0,345687	0,311209	0,384627
1300	0,356605	0,363386	0,327967	0,400516
1400	0,374914	0,381261	0,346320	0,419568
1500	0,386905	0,394279	0,360077	0,433928
1600	0,400441	0,408772	0,375837	0,446537
1700	0,417357	0,424908	0,393609	0,459355
1800	0,437291	0,442877	0,412122	0,474190
1900	0,459477	0,465017	0,433517	0,488958
2000	0,484573	0,485985	0,455085	0,503984
2250	0,535531	0,537305	0,509789	0,538064
2500	0,586808	0,584540	0,554040	0,571080
2750	0,627279	0,627757	0,599259	0,601708
3000	0,656902	0,657275	0,628713	0,623868
3500	0,704324	0,700110	0,679342	0,664491
4000	0,736108	0,729860	0,716995	0,695693
4500	0,758132	0,750143	0,742533	0,716511
5000	0,779494	0,772824	0,764298	0,734751
6000	0,813839	0,805895	0,800861	0,764265
7000	0,841438	0,833298	0,830392	0,791838
8000	0,861133	0,853104	0,852653	0,821170
10000	0,891472	0,886330	0,883843	0,856627
12000	0,909781	0,905045	0,902937	0,877572
14000	0,923803	0,919016	0,917649	0,893481
16000	0,935020	0,930630	0,928186	0,910701
18000	0,943334	0,939965	0,936337	0,921452
20000	0,949256	0,946640	0,943070	0,929572
22000	0,955392	0,953361	0,949462	0,936250
26000	0,964748	0,962205	0,958336	0,946136
30000	0,970539	0,968153	0,964957	0,953581
34000	0,975751	0,973607	0,970477	0,961247
38000	0,979199	0,976985	0,974285	0,969801
42000	0,982137	0,979818	0,977276	0,974929
46000	0,984259	0,982650	0,980300	0,979673
50000	0,986037	0,984519	0,982408	0,982266
60000	0,989313	0,988474	0,986232	0,986457
70000	0,992069	0,991044	0,988988	0,989323
80000	0,993925	0,992891	0,991125	0,991282
90000	0,995167	0,994188	0,992623	0,992763
100000	0,995907	0,995112	0,993721	0,993881

TAB. 7.3: Valores X e Y das curvas presentes na FIG. 4.4.

X	Y ISP-T1	Y ISP-T2	Y ISP-T3	Y ACD-T1
0,0	0,057620	0,063995	0,044859	0,217249
0,1	0,200003	0,219224	0,185089	0,455576
0,2	0,390196	0,399851	0,357475	0,710258
0,3	0,538815	0,563142	0,532560	0,796432
0,4	0,631443	0,652906	0,636617	0,846879
0,5	0,707260	0,723596	0,712196	0,879170
0,6	0,762695	0,776325	0,769527	0,902773
0,7	0,808144	0,818196	0,814508	0,918367
0,8	0,846155	0,852232	0,850484	0,930719
0,9	0,877804	0,880479	0,880085	0,940565
1,0	0,902581	0,903062	0,903744	0,948118
1,1	0,920709	0,920022	0,921376	0,953290
1,2	0,931200	0,931214	0,933408	0,956883
1,3	0,939129	0,939092	0,941809	0,959304
1,4	0,945379	0,945415	0,948796	0,961068
1,5	0,950715	0,950804	0,954479	0,962420
1,6	0,955280	0,955378	0,959168	0,963471
1,7	0,959296	0,959250	0,963183	0,964344
1,8	0,962793	0,962577	0,966645	0,965041
1,9	0,965835	0,965524	0,969594	0,965647
2,0	0,968554	0,968152	0,972177	0,966159
2,2	0,972918	0,972185	0,976319	0,966951
2,4	0,976123	0,975401	0,979417	0,967531
2,6	0,978630	0,977872	0,981877	0,968052
2,8	0,980737	0,979914	0,983753	0,968483
3,0	0,982589	0,981902	0,985644	0,968917
3,2	0,984278	0,983639	0,987052	0,969316
3,4	0,985875	0,985313	0,988288	0,969692
3,6	0,987248	0,986695	0,989368	0,969980
3,8	0,988378	0,987833	0,990281	0,970224
4,0	0,989362	0,988811	0,991055	0,970423
4,2	0,990131	0,989635	0,991668	0,970588
4,4	0,990777	0,990327	0,992230	0,970759
4,6	0,991333	0,990954	0,992741	0,970906
4,8	0,991881	0,991535	0,993179	0,971048
5,0	0,992368	0,992033	0,993595	0,971154
5,5	0,993398	0,993073	0,994417	0,971450
6,0	0,994255	0,993954	0,995368	0,971655
6,5	0,994929	0,994626	0,995904	0,971871
7,0	0,995422	0,995162	0,996350	0,972059
7,5	0,995842	0,995637	0,996687	0,972201
8,0	0,996220	0,996021	0,996971	0,972359
8,5	0,996518	0,996391	0,997228	0,972580
9,0	0,996810	0,996661	0,997457	0,973250
9,5	0,997073	0,996945	0,997706	0,975775
10,0	0,997287	0,997174	0,997903	0,988541

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo