Modelo de armazenamento de fluxos de rede para análises de tráfego e de segurança

Jorge Luiz Corrêa[1], André Proto[1*], Adriano Mauro Cansian[1]

[1]Instituto de Biociências, Letras e Ciências Exatas (IBILCE) – Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)
Campus de São José do Rio Preto – SP – Brasil

{jorge,andreproto,adriano}@acmesecurity.org

Abstract. *The IPFIX standard, increasingly used by network managers, allows the traffic analysis and tracking of large-scale computer networks. Its analysis methodology consumes low computational cost if compared to packet analysis methodology. The IPFIX provides a set of specifications to summarize network information, but not prevents a storage model of this information. The purpose of this article is to propose a storage model which uses relational database, to be used as standard for applications based on flow analysis. These applications will use the versatility that relational databases provide for data manipulation. Also, the resources of structured query language (SQL) enable traffic and security analysis with large precision.*

Resumo. *O padrão IPFIX, cada vez mais utilizado por administradores de rede, permite a análise e monitoramento do tráfego de uma rede de computadores de larga escala. Suas metodologias de análise consomem baixo custo computacional se comparadas à metodologia de análise de pacotes. O IPFIX fornece uma série de especificações para a sumarização de informações da rede, mas não prevê um modelo de armazenamento dessas informações. O objetivo deste artigo é propor um modelo de armazenamento utilizando banco de dados relacionais que seja uma base para aplicações voltadas à análise de fluxos. Essas usufruirão da versatilidade na manipulação de dados que um banco relacional oferece. Além disso, os recursos da linguagem SQL possibilitam análises de tráfego e de segurança de grande precisão.*

1. Introdução

1.1. Motivação e objetivos

O desenvolvimento de aplicações que utilizam redes de computadores é crescente nos dias atuais. Aplicações multimídia, distribuídas, comunicadores instantâneos, entre outras, juntamente com o número crescente de usuários, contribui vertiginosamente para o crescimento da Internet.

O grande problema para os administradores de redes é como lidar com a análise e monitoramento do tráfego de sua rede de forma escalável. Muitos sistemas desenvolvidos para este fim utilizam a técnica de captura de pacotes, analisando seus conteúdos, gerando dados estatísticos e detectando eventos de segurança. Porém, para uma rede de grande porte que possua grande quantidade de dispositivos interligados

[*] Segundo autor financiado pela FAPESP, número do processo 2007/06138-3.

esta técnica se torna inviável, visto que a captura de pacotes demandará grande processamento e espaço para armazenamento dos dados.

Devido à necessidade de protocolos que ofereçam informações do tráfego de uma rede de grande porte com baixo custo computacional, o IETF (*Internet Engineering Task Force – Internet Society*), órgão regulador de padrões para Internet, propôs a criação do padrão IPFIX (IP Flow Information Export) [Quittek, Zseby, Claise e Zander 2004] cuja finalidade era estabelecer uma arquitetura para análise de tráfego. Subseqüentemente a este trabalho, diversos protocolos foram propostos, dentre eles o *NetFlow* RFC 3954 [Claise 2004], criado pela *Cisco Systems*. O grupo de trabalho escolheu o *NetFlow* como protocolo a ser desenvolvido e implementado.

O padrão *IPFIX* é cada vez mais utilizado pelos administradores para análise e monitoramento do tráfego de suas redes. Ele fornece uma série de especificações para a sumarização de informações da rede, possibilitando de forma ampla a análise do tráfego, incluindo a detecção de eventos de segurança [Cansian e Corrêa, 2007]. Porém o RFC 3917 que descreve o padrão não prevê um modelo de armazenamento dessas informações, ficando a cargo dos desenvolvedores de aplicações o modo de armazenamento desses dados. Com isso, problemas como a impossibilidade de uma aplicação utilizar dados armazenados por outra aplicação e o baixo desempenho da solução de armazenamento escolhida por um desenvolvedor interferem na eficiência da análise de tráfego e detecção de eventos de segurança.

O objetivo deste trabalho é propor um modelo de armazenamento dos fluxos de rede provindos do padrão IPFIX (protocolo *NetFlow*), possibilitando a criação de uma infra-estrutura para aplicações voltadas à análise desses fluxos. O trabalho baseia-se no modelo de banco de dados relacional e, utilizando recursos oferecidos pelo Sistema Gerenciador de Banco de Dados e pela linguagem estruturada de consulta (SQL), desenvolve diversas técnicas de consulta aos dados que possibilitam analisar e monitorar o tráfego de uma rede além de detectar eventos de segurança.

1.2. Trabalhos relacionados

São vários os trabalhos e aplicações que criaram seus próprios modelos de armazenamento de fluxos, como as ferramentas proprietárias de [Adventnet 2007] [Networks 2008]. O software *flow-tools* [Fullmer 2007], um dos mais utilizados por administradores pelo fato de ser um software livre, armazena os fluxos do *NetFlow* em vários arquivos diferenciados por tempo. Uma de suas ferramentas denominada *flow-scan* gera gráficos a partir desses arquivos, armazenando somente os dados utilizados no gráfico no formato RRD [Dave 2000]. O armazenamento dos fluxos do *NetFlow* em arquivos possibilita ao *flow-tools* a compactação dos mesmos, otimizando o espaço utilizado em disco. Porém utilizando este recurso perde-se desempenho no processamento das informações, visto a necessidade de descompactá-los. Outra desvantagem do *flow-tools* é a impossibilidade de realizar *cache* dos fluxos na memória, o que aumentaria o desempenho no acesso aos dados; além disso, aplicações que necessitarem dos dados do *NetFlow* terão que utilizar o próprio *flow-tools* para acessá-los, ficando dependente dessa ferramenta.

O trabalho de [Bill 2000] propõe o armazenamento do *NetFlow* versão 5 em um banco de dados relacional para gerar dados estatísticos e detectar eventos de segurança. Ele demonstra que a tecnologia de banco de dados relacional é uma boa solução para o

problema de armazenamento. Nele há uma proposta de um modelo de tabela para o banco relacional e algumas consultas que identificam eventos nos dados do *NetFlow*, utilizando um supercomputador para armazenar e consultar os dados. O trabalho deste artigo propõe melhorias no modelo proposto por [Bill, 2000] tais como: otimização no espaço utilizado para armazenamento; modelo de várias tabelas para buscas otimizadas; consultas detalhadas para detectar diversos tipos de eventos na rede.

2. Conceitos gerais

2.1. Fluxos de redes

A Cisco define um fluxo de redes como uma seqüência unidirecional de pacotes entre máquinas de origem e destino. Pode-se dizer em resumo que o *NetFlow* provê a sumarização de informações sobre o tráfego de um roteador ou switch. Fluxos de rede são altamente granulares; eles são identificados por ambos os endereços IP bem como pelo número das portas da camada de transporte. O *NetFlow* também utiliza, para identificar unicamente um fluxo, os campos "*Protocol type*" e "*Type of Service*" (*ToS*) do IP e a interface lógica de entrada do roteador ou switch. Os fluxos mantidos no *cache* do roteador/switch são exportados para um coletor nas seguintes situações: permanece ocioso por mais de 15 segundos; sua duração excede 30 minutos; uma conexão TCP é encerrada com a *flag* FIN ou RST; a tabela de fluxos está cheia ou o usuário redefine as configurações de fluxo. É importante notar que o tempo máximo que um fluxo permanece no dispositivo antes de ser exportado é de 30 minutos.

A Figura 1 ilustra os campos do protocolo *NetFlow*, bem como o seu cabeçalho. Os campos que realmente interessam neste trabalho estão descritos em *"Flow Record Format"*. Eles são responsáveis por representar as informações sumarizadas de uma conexão/sessão entre duas máquinas, descrevendo endereços de origem e destino, portas de origem e destino, interfaces de entrada e saída do roteador, número de pacotes e octetos envolvidos, *timestamp* de criação do fluxo e *timestamp* de sua última atualização (campos *first* e *last*), flags do TCP, entre outros.

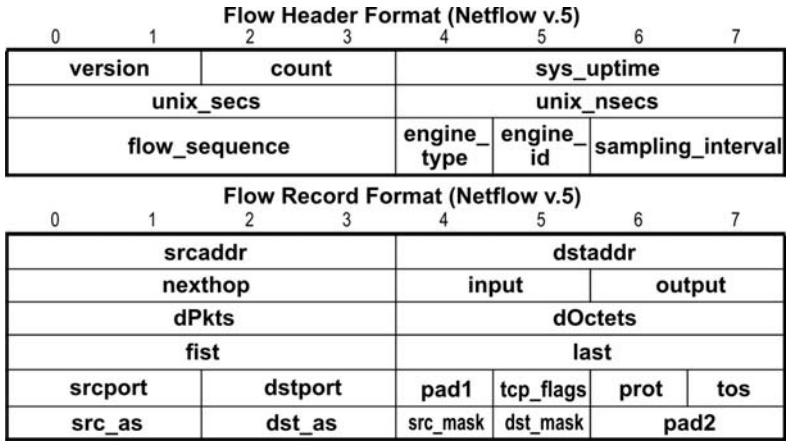

Figura 1. Formato de um datagrama *NetFlow*.

2.2. Modelo de dados relacional

Em [Elmasri e Navathe 2005] é descrito o modelo relacional como a representação do banco de dados como uma coleção de relações. Cada relação se

apresenta como uma tabela, em que cada linha na tabela representa uma coleção de valores e é conhecida como *tupla*; já as colunas são conhecidas como *atributos* que identificam a correta interpretação de cada valor disposto nas linhas.

No modelo proposto por este artigo as colunas representam os campos do protocolo *NetFlow* e as linhas representam os fluxos gerados pelo dispositivo de rede. Cada linha (tupla) é um fluxo unidirecional sumarizando uma conexão/sessão.

3. O modelo de armazenamento

Esta seção descreve as principais características do modelo de armazenamento proposto por este artigo. Como citado anteriormente, o modelo de armazenamento é baseado no modelo de banco de dados relacionais, utilizando tabelas para armazenar os dados. Abaixo são dispostas as características do modelo:

- Armazenamento de variáveis: os campos que o *NetFlow* descreve devem ser armazenados em variáveis de tamanho correspondente ao tamanho descrito pelo protocolo, em bytes. Como exemplo, IPs são armazenados em inteiros de 4 bytes. Campos do *NetFlow* desnecessários para uma determinada rede podem ser ignorados (decisão a ser tomada pelo administrador);
- Tabelas periódicas: As tabelas devem conter fluxos de redes de determinado período de tempo. Sugere-se que cada tabela armazene um dia de fluxo (24 horas) e que possua como nome a data na qual correspondem os fluxos. Essa característica é importante, pois em redes com grande volume de tráfego o número de tuplas inseridas na tabela é extremamente grande e, se for utilizado uma única tabela para armazenar todas as tuplas, a realização de consultas aos dados ficariam inviáveis;
- Tabelas adicionais:

  – Tabela de fluxos atuais: A criação de uma tabela contendo os fluxos dos 30 minutos atuais provê maior acessibilidade aos dados e geração de estatísticas. Esta tabela permitirá consultas que detectem eventos recentes na rede (como ataques a determinadas máquinas) com alto desempenho visto seu menor número de tuplas. Um fluxo *Netflow* chega ao coletor de forma desordenada (ordem cronológica), mas permanece no dispositivo de rede por no máximo 30 minutos antes de ser exportado [Claise 2004]; assim a tabela de 30 minutos auxilia na ordenação dos dados, pois os fluxos mais antigos que este período serão retirados desta tabela e inseridos na respectiva tabela periódica de forma ordenada com a garantia de não existir um fluxo fora de ordem a ser exportado.

  – Tabela de representação unidirecional: Sugere-se a criação de tabelas que contenham os fluxos representando o tráfego unidirecional entre duas máquinas. Uma tabela denominada “input” registra os fluxos que representam tráfego externo para interno e uma tabela denominada “output” registra os fluxos de modo inverso. Estas tabelas são úteis para cruzar informações e reconstituir conexões/sessões entre duas máquinas de forma bidirecional.

  – Tabela de reconstituição de conexões/sessões: Esta tabela é gerada a partir das tabelas de representação unidirecional, reconstituindo conexões/sessões de forma bidirecional. Com ela é possível, por exemplo, verificar o total de pacotes

e bytes envolvidos em uma conexão/sessão. Ela deverá ser construída através do produto cartesiano entre as tabelas “input” e “output”.

- Sistema Gerenciador de Banco de Dados (SGBD): A escolha do SGBD [Elmasri e Navathe 2005] a ser utilizado deverá seguir os seguintes critérios:
  - Suporte a SQL (Struct Query Language) [Elmasri e Navathe 2005];
  - A inserção de tuplas no banco deve ser mais rápida que a velocidade de geração dos fluxos pelo roteador/switch. Em [Bill 2000] recomenda-se utilizar um SGBD que possa inserir de forma três vezes mais rápida que a geração de fluxos em tempo real, isto porque será necessário sobrar recursos do SGBD para a realização das consultas;
  - Suporte a indexação de colunas (visa aumentar o desempenho das consultas).

O modelo aqui proposto traz apenas características iniciais para o armazenamento dos fluxos. Ele não impede que outros recursos sejam utilizados caso sejam úteis à base de dados como, por exemplo, a clusterização das tabelas e a fusão de tabelas utilizando índices. Esses recursos não serão abordados neste artigo.

4. A implementação do modelo

As características principais do modelo não foram definidas por acaso; vários testes foram realizados em busca da melhor solução para armazenamento dos fluxos. Esta seção descreve a implementação realizada e algumas considerações que levaram ao modelo proposto. Foi utilizada a versão 5 do protocolo *NetFlow* devido às restrições do ambiente de teste.

4.1. Tabelas

4.1.1 Formato das tabelas

A criação das tabelas foi inicialmente baseada em [Bill 2000]. A tabela proposta por [Bill 2000] está descrita abaixo:

Tabela 1. Formato da tabela proposto por [Bill 2000].

Campos	Tipo de dados	Representação	Tamanho
Router_id	char(1)	not null,	1 byte
Srcaddr	Bigint	unsigned not null,	8 bytes
Dstaddr	Bigint	unsigned not null,	8 bytes
Nexthop	Bigint	unsigned not null,	8 bytes
Input	Smallint	unsigned not null,	2 bytes
Output	Smallint	unsigned not null,	2 bytes
Packets	Integer	unsigned not null,	4 bytes
Octets	Integer	unsigned not null,	4 bytes
First	Timestamp	not null,	4 bytes
Last	Timestamp	not null,	4 bytes
Srcport	Smallint	unsigned not null,	2 bytes
Dstport	Smallint	unsigned not null,	2 bytes
Tcp_flags	Tinyint	unsigned not null,	1 byte
Prot	Tinyint	unsigned not null,	1 byte
Tos	Tinyint	unsigned not null,	1 byte
Srcas	Smallint	unsigned not null,	2 bytes
Dstas	Smallint	unsigned not null,	2 bytes
Srcmask	Tinyint	unsigned not null,	1 byte
Dstmask	Tinyint	unsigned not null	1 byte

Somando-se o tamanho das variáveis na Tabela 1, tem-se que cada fluxo *NetFlow* consome 58 bytes de espaço em disco. Outro detalhe importante é que os campos "*srcaddr*", "*dstaddr*" e "*nexthop*", que representam IPs, são armazenados de forma literal, semelhante a uma *string*. Como exemplo, o IP 192.168.20.1 seria escrito como um inteiro 192168020001 de 8 bytes.

Sabe-se que o IP é formado por 4 bytes. Assim a proposta deste artigo se concentrou em otimizar os campos que representam IP, utilizando variáveis inteiras de 4 bytes para gravar os campos "*srcaddr*", "*dstaddr*" e "*nexthop*". Como não é possível armazenar um IP em notação semelhante à de [Bill 2000] utilizando este tipo de variável, estes então são armazenados na forma de inteiros de 32 bits, ou seja, os quatro octetos são unidos resultando em um inteiro abstrato. Como exemplo, o IP 192.168.20.1 seria escrito como inteiro 3232240641. Assim, pela proposta deste artigo, as colunas da tabela do banco de dados serão criadas conforme Tabela 2:

Tabela 2. Formato da tabela proposto por este trabalho.

Campos	Tipo de dados	Representação	Tamanho
Router_id	char(1)	not null,	1 byte
Srcaddr	Integer	unsigned not null,	4 bytes
Dstaddr	Integer	unsigned not null,	4 bytes
Nexthop	Integer	unsigned not null,	4 bytes
Input	Smallint	unsigned not null,	2 bytes
Output	Smallint	unsigned not null,	2 bytes
Packets	Integer	unsigned not null,	4 bytes
Octets	Integer	unsigned not null,	4 bytes
First	Timestamp	not null,	4 bytes
Last	Timestamp	not null,	4 bytes
Srcport	Smallint	unsigned not null,	2 bytes
Dstport	Smallint	unsigned not null,	2 bytes
tcp_flags	Tinyint	unsigned not null,	1 byte
Prot	Tinyint	unsigned not null,	1 byte
Tos	Tinyint	unsigned not null,	1 byte
Srcas	Smallint	unsigned not null,	2 bytes
Dstas	Smallint	unsigned not null,	2 bytes
Srcmask	Tinyint	unsigned not null,	1 byte
Dstmask	Tinyint	unsigned not null	1 byte

Com isso cada fluxo *Netflow* consumirá 46 bytes de armazenamento: uma economia de aproximadamente 20% em relação a proposta de [Bill 2000].

Uma questão que fica em aberto é como obter novamente o IP a partir de um inteiro abstrato. Será visto adiante que isso é possível utilizando os recursos que o SGBD oferece.

4.1.2. Estrutura de tabelas no banco de dados

Conforme modelo proposto, as tabelas são criadas para armazenarem um dia de fluxos de redes. A nomenclatura da tabela foi definida como ***afmYYYYMMDD***, sendo YYYY, MM e DD, o ano, mês e dia correntes, respectivamente. Além disso, mais quatro tabelas foram criadas:

- *Last30minutes:* Tabela que contém fluxos dos últimos 30 minutos. Esta tabela é criada na memória para obter maior desempenho em seu acesso;
- *Input:* Grava os fluxos que representam conexões/sessões de fora para dentro da rede em análise, dos últimos 5 minutos.

- *Output:* Grava os fluxos que representam conexões/sessões de dentro para fora da rede em análise, dos últimos 5 minutos.
- *Connections_Sessions:* Tabela que faz o produto cartesiano *Input* x *Output*, reconstituindo as conexões/sessões de modo bidirecional.

4.2. Coletor de fluxos

Os fluxos exportados pelo roteador/switch são coletados por um *socket* UDP em uma máquina denominada "coletora". O *socket* foi implementado em linguagem JAVA, seguindo as especificações do *NetFlow* versão 5 [Systems 2004]. O coletor utiliza três *threads* que desempenham características importantes a serem citadas:

- O primeiro *thread* desempenha o papel de coletor, recebendo os datagramas UDP e armazenando os dados úteis em um buffer;
- O segundo *thread* retira os dados do buffer, seleciona os campos que interessa (definidos pelo administrador) e armazena na tabela *last30minutes* do banco de dados;
- O terceiro *thread* gerencia os fluxos nas tabelas, criando as tabelas diárias e transferindo os fluxos mais antigos que 30 minutos da tabela *last30minutes* para a tabela de data correspondente ao fluxo; isso porque há a certeza de não existirem fluxos mais antigos que 30 minutos a serem exportados pelo roteador. Assim esse *thread* inserirá nas tabelas diárias os fluxos em ordem cronológica, facilitando a indexação das tabelas (coluna *first*).

4.3. Sistema Gerenciador de Banco de Dados

Dentre os vários Sistemas Gerenciadores de Banco de Dados existentes, aqueles que são softwares livres e mais utilizados são MySQL [Mysql 2008] e Oracle [Oracle 2008]. O trabalho de [Bill 2000] descreve várias vantagens do MySQL em relação ao Oracle. Seguindo as descrições feitas por [Bill 2000], o SGBD escolhido foi o MySQL.

O MySQL proporciona vários recursos que auxiliam a consulta aos dados. Um exemplo disso são as funções *inet_ntoa()* e *inet_aton()* que convertem números decimais em uma expressão alfa-numérica no formato do IPv4, e vice-versa, resolvendo a questão levantada na seção 4.1.1. Como exemplo, para obter o IP do número decimal 3232240641, basta inseri-lo na função: *inet_ntoa(3232240641) = 192.168.20.1*.

4.4. Ambiente

O ambiente de testes foi montado em uma universidade e conta com mais de 1000 dispositivos de rede incluindo máquinas, roteadores e dispositivos móveis. O ambiente possui um roteador CISCO 7200 VXR que exporta fluxos *NetFlow* versão 5. A máquina coletora é um PC x86 Pentium IV 1.8GHz, 768MB RAM e HD IDE 80GB ATA-100, dedicado a coleta, armazenamento e análise dos fluxos no banco (Figura 2).

No ambiente citado, a quantidade de fluxos de rede gerados pelo roteador está em torno de 10.000.000 de fluxos/dia. Isso significa que, em média, 115 tuplas são inseridas por segundo no banco de dados. Um dos maiores picos de inserção na tabela registrados foi de 2000 tuplas em um segundo. Testes foram realizados e mostraram que

o MySQL inseriu 10000 tuplas em 0.07 segundos, e 295563 tuplas em 1.55 segundo. Este desempenho é mais que satisfatório para inserção de dados no banco.

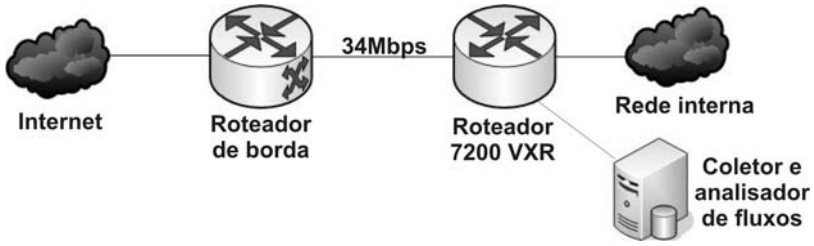

Figura 2. Representação da topologia do ambiente.

5. Consultas e resultados

Esta seção descreve as consultas aos dados que resultam em análises de tráfego e de segurança. É importante observar que dados como IP de origem e destino nas consultas foram sanitizados, a fim de preservar a privacidade de máquinas e usuários. A sintaxe para consultas SQL obedece ao seguinte padrão:

SELECT nomes_das_colunas FROM nome_da_tabela [WHERE condições];

5.1. Medição de tráfego

As três consultas a seguir medem a quantidade de fluxos gerados em um dia, para o tráfego total (I), de entrada (II) e de saída da rede (III).

I. *SELECT count(*) AS Total FROM afm20080311;*

II. *SELECT count(*) AS Input FROM afm20080311 WHERE input=28;*

III. *SELECT count(*) AS Output FROM afm20080311 WHERE output=28;*

```
+----------+            +---------+              +---------+
| Total    |            | Input   |              | Output  |
+----------+            +---------+              +---------+
| 10756345 |            | 5215475 |              | 5227989 |
+----------+            +---------+              +---------+
1 row in set (0.03 sec) 1 row in set (21.89 sec) 1 row in set (4.67 sec)
```

Figura 3. Resultados das consultas I, II e III.

A consulta I é realizada em menos de um segundo, isto porque o SGBD possui índices internos que contabilizam o número de tuplas em uma tabela. Já na consulta II é necessário percorrer as tuplas verificando se essas atendem as condições; assim a consulta leva mais de 20 segundos para ser concluída. Nota-se que a consulta III realiza a mesma verificação da II, porém é resolvida em tempo menor. A justificativa é que, quando a consulta II é executada, um *cache* é criado na memória contendo as tuplas consultadas; assim quando se executa a consulta III logo após a II, o *cache* é utilizado resultando em uma consulta mais rápida.

As consultas a seguir fazem a contagem de fluxos que sumarizam uma sub-rede específica (tráfego de entrada) pertencente ao ambiente local (IV) e a contagem de fluxos que sumarizam dados de um único host (V):

IV. *SELECT count(*) AS Input_Network FROM afm20080311 WHERE inet_ntoa(dstaddr) like 192.168.202.%%%' and input=28;*

V. SELECT count(*) AS Input_Host FROM afm20080311 WHERE dstaddr=inet_aton('192.168.202.5') and input=28;

```
+---------------+          +------------+
| Input_Network |          | Input_Host |
+---------------+          +------------+
|        476350 |          |      21395 |
+---------------+          +------------+
1 row in set (9.83 sec)    1 row in set (6.40 sec)
```

Figura 4. Resultado das consultas IV e V.

A consulta a seguir conta os fluxos por minuto (no período de um dia) e realiza uma média de fluxos/segundo, resultando em dados estatísticos:

VI. SELECT date_sub(first, interval second(first) second) AS Date, count(*) AS Flows, count(*)/60 AS Flows_per_second FROM afm20080311 WHERE input=28 GROUP BY hour(first),minute(first) ORDER BY first;

```
+---------------------+-------+-----------------+
| Date                | Flows | Flows_per_second |
+---------------------+-------+-----------------+
| 2008-03-11 00:00:00 |  2492 |          41.5333 |
| 2008-03-11 00:01:00 |  2037 |          33.9500 |
| 2008-03-11 00:02:00 |  2396 |          39.9333 |
| 2008-03-11 00:03:00 |  2196 |          36.6000 |
| 2008-03-11 00:04:00 |  2146 |          35.7667 |
| 2008-03-11 00:05:00 |  2311 |          38.5167 |
|         ...         |   ... |             ...  |
| 2008-03-11 23:58:00 |  2416 |          40.2667 |
| 2008-03-11 23:59:00 |  2095 |          34.9167 |
+---------------------+-------+-----------------+
1440 rows in set (21.36 sec)
```

Figura 5. Resultado da consulta VI.

Com essa consulta já é possível gerar gráficos que mostrem o comportamento do tráfego da rede, como na Figura 6. Além disso, o resultado dessa consulta serve como dados de entrada para detectores de intrusão por anomalia, como em [Cansian e Corrêa 2007] que utiliza fluxos de rede *NetFlow* e redes neurais para detectar anomalias em um tráfego na rede.

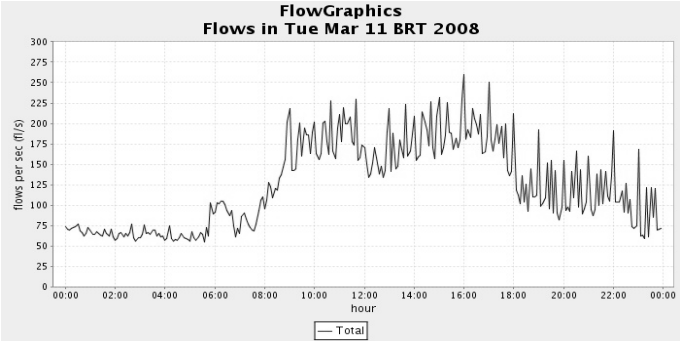

Figura 6. Gráfico gerado por aplicação JAVA utilizando resultados de VI.

A consulta a seguir retorna as máquinas que mais receberam tráfego (em bytes) nos últimos 5 minutos. Realizando a consulta na tabela *last30minutes*, a resposta é obtida em menos de um segundo.

VII. SELECT inet_ntoa(dstaddr) AS Destination,sum(dPkts)/300 AS pkts_per_sec_in, sum(dOctets)/300 AS bytes_per_sec_in FROM last30minutes WHERE input=28

and first > date_sub(now(),interval 5 minute) GROUP BY dstaddr ORDER BY sum(dOctets) DESC LIMIT 10;

Destination	pkts_per_sec_in	bytes_per_sec_in
192.168.210.173	274.9367	410820.0633
192.168.209.39	83.0467	119261.5900
192.168.205.81	27.5567	40465.0667
192.168.210.150	35.4333	34726.8833
192.168.204.171	19.9500	28596.5667
192.168.203.33	16.8767	24871.6967
192.168.205.190	17.3267	24237.5000
192.168.209.36	21.3233	23427.2300
192.168.205.79	25.2867	21427.6867
192.168.201.91	16.5100	13426.5433

10 rows in set (0.49 sec)

Figura 7. Resultado da consulta VII.

A consulta a seguir retorna as máquinas que mais geraram fluxos de saída nos últimos 5 minutos. Esta consulta permite identificar máquinas que geram tráfego anômalo. Isso porque, como um fluxo de rede representa unidirecionalmente uma conexão/sessão, um host que gerar uma grande quantidade de fluxos indicará grande quantidade de conexões ou sessões, comportamento comum de worms [Cert.br 2006], prospecção na rede (*portscan*) [Fyodor 1997], *compartilhamento de arquivos* (*file sharing*) entre outros.

VIII. SELECT inet_ntoa(srcaddr) AS Source, count(srcaddr) AS Flows, count(srcaddr)/300 AS Flows_per_sec FROM last30minutes WHERE first > date_sub(now(), interval 5 minute) and output=28 GROUP BY srcaddr ORDER BY flows DESC LIMIT 10;

Source	Flows	Flows_per_sec
192.168.212.236	3232	10.7733
192.168.201.1	2460	8.2000
192.168.210.150	1898	6.3267
192.168.205.79	1477	4.9233
192.168.202.163	1215	4.0500
192.168.201.91	1148	3.8267
192.168.205.235	565	1.8833
192.168.202.9	525	1.7500
192.168.201.11	394	1.3133
192.168.207.164	391	1.3033

10 rows in set (0.41 sec)

Figura 8. Resultado da consulta VIII.

A inserção a seguir consulta as tabelas *Input* e *Output* para o cruzamento de informações através de um produto cartesiano de forma a reconstituir conexões/sessões. O resultado é inserido na tabela *Connections_Sessions* e pode ser visto na Figura 9. Como visto na Figura 9, vários dados foram reconstituídos: IPs da rede interna e externa, portas de origem e destino, pacotes enviados e recebidos, bytes trafegados, protocolo utilizado e flags TCP envolvidas.

IX. INSERT INTO connections_sessions (insidehost, outsidehost, dPktsIn, dPktsOut, dOctetsIn, dOctetsOut, first, last, insideport, outsideport, tcp_flagsIn, tcp_flagsOut, prot) SELECT input.destination, input.source, input.dPkts,

output.dPkts, input.dOctets, output.dOctets, least(input.first,output.first), greatest(input.last,output.last), input.dstport, input.srcport, input.tcp_flags, output.tcp_flags, input.prot FROM input, output WHERE input.source = output.destination and input.destination = output.source and input.srcport = output.dstport and input.dstport = output.srcport;

InsideHost	OutsideHost	insideport	outsideport	tcp_flagsIn	tcp_flagsOut	prot	dPktsIn	dPktsOut	dOctetsIn	dOctetsOut
192.168.201.1	10.23.59.51	53	53	16	16	17	5	5	1185	445
192.168.201.91	10.23.59.124	1191	80	27	30	6	14	11	15461	1437
192.168.201.91	10.23.59.124	2029	80	27	30	6	18	16	19535	2291
192.168.201.91	10.23.59.215	4406	80	27	30	6	15	11	16215	1406
192.168.201.91	10.23.59.215	4418	80	27	30	6	23	14	26602	1717
192.168.202.51	10.23.59.215	4426	80	27	30	6	7	7	5928	1246
192.168.202.5	10.23.59.105	1140	80	27	30	6	15	12	15761	1697
192.168.202.5	10.23.59.105	1166	80	27	30	6	19	14	20826	1805
192.168.203.12	10.23.55.61	1603	80	27	30	6	10	8	8514	765
192.168.203.12	10.23.55.61	1766	80	27	30	6	12	11	11045	1571

10 rows in set (0.00 sec)

Figura 9. Tabela *Connections_Sessions*.

5.2. Detecção de intrusão

As consultas desta seção identificam eventos relacionados à detecção de ataques e utilização de serviços indesejáveis como *compartilhamento de arquivos*. A consulta a seguir identifica máquinas externas realizando prospecção na rede interna (*portscan*). Para identificar esse tipo de evento é necessário filtrar padrões do mesmo, como a flag SYN do TCP aliada à curta duração da conexão, identificados nos campos *tcp_flags*, *first* e *last* do *NetFlow*.

X. SELECT inet_ntoa(srcaddr) AS Source, count(distinct dstaddr) AS Hosts_scanned, count(distinct dstport) AS Ports_scanned FROM last30minutes WHERE input="28" and tcp_flags & 2 = "2" and timediff(last,first) < "00:00:15" GROUP BY srcaddr HAVING ports_scanned > 300 or hosts_scanned > 300 ORDER BY hosts_scanned;

Source	Hosts_scanned	Ports_scanned
10.208.77.63	1	301
10.228.167.216	1	371
10.54.170.30	3	469
10.234.200.22	3	308
10.225.157.30	9	1042
10.176.3.142	11	394
10.142.228.136	15	953
10.221.6.19	22	768
10.109.112.136	22	985
10.152.160.212	28	1457
10.108.13.126	62	317
192.168.1.9	83	484
10.108.13.122	92	640
192.168.2.109	175	4644
10.82.33.10	331	7

15 rows in set (1.35 sec)

Figura 10. Resultado da consulta X.

A consulta a seguir identifica máquinas externas realizando ataque de dicionário no serviço SSH das máquinas da rede interna. Os padrões deste ataque são semelhantes ao anterior, acrescentando o fato das conexões serem na porta 22 e direcionadas a um

único host interno. Na consulta são selecionadas apenas as máquinas que efetuaram mais de 100 tentativas em uma mesma máquina de destino.

XI. *SELECT inet_ntoa(srcaddr) AS Source,inet_ntoa(dstaddr) AS Destination, count(*) AS Attempts FROM afm20080311 WHERE dstport="22" and tcp_flags & 2 ="2" and inet_ntoa(dstaddr) like "192.168.%%%.%%%" and timediff(last,first) < "00:00:15" GROUP BY srcaddr,dstaddr HAVING attempts > 100 ORDER BY attempts;*

Source	Destination	Attempts
10.64.116.10	192.168.202.5	120
10.49.167.66	192.168.203.15	124
10.129.152.84	192.168.203.115	152
10.129.152.84	192.168.201.15	152
10.49.167.66	192.168.216.2	155
10.129.152.84	192.168.216.2	169
		...
10.254.106.211	192.168.203.230	2246
10.254.106.211	192.168.202.105	2281

29 rows in set (19.04 sec)

Figura 11. Resultado da consulta XI.

A consulta a seguir identifica possíveis máquinas da rede interna utilizando aplicativos *compartilhadores de arquivos*. Parte-se do principio que eventos deste tipo possuem várias conexões em portas altas (maiores que 1024) tanto para IP de origem quanto para IP de destino, além do número considerável de bytes recebidos e enviados.

XII. *SELECT inet_ntoa(dstaddr) AS Destination, count(distinct srcaddr) AS Num_Sources_Host, sum(dOctets) AS Bytes, sum(dOctets)/ time_to_sec(timediff(now(), date_sub(now(), interval 30 minute))) AS Bytes_per_second FROM last30minutes WHERE dstport>1024 and srcport>1024 and input=28 GROUP BY dstaddr HAVING Num_Sources_Host > 100 and Bytes > 1000000 ORDER BY Bytes;*

Destination	Num_Sources_Host	Bytes	Bytes_per_second
192.168.203.200	149	1428651	793.6950
192.168.212.104	150	1788306	993.5033
192.168.212.236	7663	2288412	1271.3400
192.168.205.235	3020	5739611	3188.6728
192.168.205.79	2691	6942263	3856.8128
192.168.210.150	3666	19184354	10657.9744

6 rows in set (0.47 sec)

Figura 12. Resultado da consulta XII.

5.3. Comparações com a ferramenta *Flow-tools*

Testes com a ferramenta *Flow-tools* foram realizados com a intenção de medir o desempenho de algumas consultas aos dados dos fluxos *NetFlow*. A Tabela 3 descreve a comparação de desempenho entre as consultas SQL proposta por este trabalho e as correspondentes no Flow-tools. Nota-se que o desempenho do *Flow-tools* é inferior as consultas SQL, isso porque o mesmo não possui recursos de gerenciamento que um SGBD possui, além de utilizar acesso direto a arquivos para armazenamento e consulta aos dados. Outros fatores importantes são a ausência do recurso de *caching* dos dados na memória para melhor desempenho das consultas e a impossibilidade de construir consultas complexas que detectam eventos como máquinas realizando prospecção *(portscan)* ou compartilhando arquivos.

A Tabela 4 descreve a comparação entre o espaço utilizado para armazenamento, com ligeiro ganho por parte do *Flow-tools* devido à compactação dos dados. Porém, ao utilizar esta técnica o *Flow-tools* perde desempenho no acesso os dados, visto que precisa descompactar os arquivos. Esse é mais um motivo para explicar o desempenho inferior visto na Tabela 3.

Tabela 3. Comparação de tempo em segundos para realizar consulta aos fluxos

Consulta	Banco de Dados	Flow-tools
Quantidade de fluxos em um dia.	0.00s	39.78s
Contagem de fluxos por minuto.	21.36s	39.78s
Tráfego de uma rede específica.	9.83s	27.52s
10 máquinas com maior número de octetos recebidos (24 horas).	30.76s	59.43s

Tabela 4. Comparação entre o espaço utilizado em disco para armazenamento.

Espaço utilizado em disco		
Consulta	**$7x10^6$ fluxos**	**1 fluxo**
Banco de dados	~300 MB	45 bytes
Flow-tools	~134 MB	~20 bytes

6. Conclusões

O protocolo *NetFlow* tem se mostrado bastante versátil e escalável quando se trata de analisar o tráfego de uma rede de computadores de grande porte. Como visto neste trabalho, o protocolo *NetFlow* armazenado em um banco de dados relacional permite melhor gerenciamento dos dados fornecidos, além de maior versatilidade em sua manipulação por quaisquer aplicações. A linguagem SQL permite a construção de consultas que ampliam a manipulação de informações, de modo a obter resultados satisfatórios em se tratando de analisar tráfego e detectar eventos de segurança.

As consultas da seção 5.1 manipulam os dados dos fluxos de forma a obter medições do tráfego de uma rede, permitindo a geração de dados estatísticos e gráficos. A consulta IX realiza o cruzamento das informações do tráfego unidirecional reconstituindo informações de tráfego na forma bidirecional. As consultas da seção 5.2 manipulam os dados dos fluxos para detectar eventos de segurança na rede, como a detecção de máquinas realizando prospecção na rede, ataques de dicionário no serviço SSH ou utilizando programas *compartilhadores de arquivos*.

Enfim, a ampla quantidade de recursos que um banco de dados relacional e seu SGBD oferecem deixa claro que não há limitações para criação de novas consultas a fim de aperfeiçoar a manipulação dos dados fornecidos pelo *NetFlow*. Isso garante que trabalhos relacionados utilizem este modelo para desenvolvimento de novas aplicações e modelos de análise de tráfego e detecção de intrusão.

7. Trabalhos futuros

O próximo passo deste trabalho terá como objetivo aperfeiçoar ainda mais o desempenho do banco de dados. Técnicas de clusterização da base de dados e processamento paralelo deverá ser o enfoque deste aperfeiçoamento. Outra iniciativa já existente é a criação de uma aplicação web para facilitar a construção de consultas e

monitorar o tráfego da rede, gerando dados estatísticos e gráficos de forma automatizada. Por fim, existe outro trabalho em desenvolvimento que utiliza consultas SQL para detecção de eventos de segurança, utilizando padrões relacionados à *malwares* e vulnerabilidades de softwares. O intuito é detectar eventos de segurança que possuam padrões de assinatura.

Referências

ADVENTNET. (2007) "ManageEngine NetFlow Analyzer" Disponível em: http://origin.manageengine.adventnet.com/products/netflow/index.html, Acesso em 10 mai. 2008.

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

CANSIAN, A. M.; CORRÊA, J. L. (2007) "Detecção de ataques de negativa de serviço por meio de fluxos de dados e sistemas inteligentes", Em VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, p. 125-141, Rio de Janeiro, RJ.

CERT.BR. (2006) "Cartilha de Segurança para Internet - Parte VIII: Códigos maliciosos (Malware) ", Disponível em: http://cartilha.cert.br/malware/. Acesso em 12 mai. 2008.

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

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

ELMASRI, R. E.; NAVATHE, S. (2005) "Sistemas de Banco de Dados", Addison-Wesley, ISBN 8588639173.

FULLMER, M. (2007) "Tool set for working with NetFlow data", Disponível em: http://www.splintered.net/sw/flow-tools/docs/flow-tools.html, Acesso em 27 dez. 2007.

FYODOR. (1997) "The Art of Port Scanning", Disponível em: http://www.insecure.org/nmap/nmap_doc.html, Acesso em 10 mai. 2008.

MYSQL. (2008) "MySQL 5.1 Reference Manual", Disponível em: http://dev.mysql.com/doc/refman/5.1/en/index.html, Acesso em 10 mai. 2008.

NETWORKS, F. (2008) "NetFlow Tracker", Disponível em: http://www.flukenetworks.com/fnet/en-us/products/NetFlow+Tracker/, Acesso em 10 mai. 2008.

ORACLE (2008). "Oracle Database Documentation", Disponível em: http://www.oracle.com/technology/documentation/database.html, Acesso em 10 mai. 2008.

QUITTEK, J., Zseby, T., Claise, B., Zander, S. (2004) "RFC 3917: Requirements for IP Flow Information Export: IPFIX", Disponível em: http://www.ietf.org/rfc/rfc3917.txt, Acesso em 27 de dez. 2007.

SYSTEMS, C. (2004) "NetFlow Packet version 5 (V5)", Disponível em: http://netflow.caligare.com/netflow_v5.htm, Acesso em: 10 mai. 2008.