Análises de fluxos para coleta de evidências

André Proto, Jorge Luiz Corrêa, Adriano Mauro Cansian, *Laboratório ACME! Computer Security Research – UNESP – Universidade Estadual Paulista "Júlio de Mesquita Filho"*

***Abstract*—The IPFIX (IP Flow Information Export) standard, increasingly used by network administrators, allows traffic analyses and tracking of large-scale computer networks, allowing evidence gathering of security events. Its analysis methodology requires lower computational cost than packet analysis methodology. The purpose of this article is to propose a storage model for IPFIX which uses relational database enabling an infrastructure for traffic analyses and intrusion detection, by means of the resources offered by structured query language (SQL). The results will provide investigation data related to events occurred on network computers.**

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

***Index Terms*—Data flow analysis, database, intrusion detection, IPFIX, NetFlow, network computers, security, SQL**

***Palavras-chave*—análise de fluxo de dados, banco de dados, detecção de intrusão, IPFIX, NetFlow, redes de computadores, segurança, SQL**

I. Introdução

A. 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, impactam diretamente nos mecanismos de segurança utilizados na 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 esta técnica é 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*) [1] cuja finalidade era estabelecer uma arquitetura para análise de tráfego. Subseqüentemente a este trabalho, diversos protocolos foram propostos, dentre eles o *NetFlow* RFC 3954 [2], 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 para análises periciais, detecção de eventos de segurança [3] e monitoramento do tráfego, pois fornece uma série de especificações para a sumarização de informações da rede. 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 pericial e na detecção de eventos de segurança.

Este trabalho tem por objetivo propor técnicas de consulta aos dados fornecidos pelo IPFIX para análise pericial e detecção de eventos de segurança, além de propor um modelo de armazenamento que provê uma infra-estrutura para aplicações voltadas à análise de fluxos. Para isso este trabalho é baseado 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), modela uma série de consulta aos dados que possibilitam analisar e monitorar o tráfego e detectar eventos de uma rede de computadores.

O primeiro autor é financiado pela FAPESP, número do processo 2007/06138-3.

A. Proto é pesquisador do Laboratório *ACME! Computer Security Research* e graduando do curso de Bacharelado em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista (andreproto@acmesecurity.org).

J. L. Corrêa é pesquisador do Laboratório *ACME! Computer Security Research* e mestrando no Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista (jorge@acmesecurity.org).

A. M. Cansian é pesquisador e coordenador do Laboratório *ACME! Computer Security Research* e Professor Doutor do Departamento de Ciências da Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista (adriano@acmesecurity.org).

B. Trabalhos relacionados

São vários os trabalhos e aplicações que criaram seus próprios modelos de armazenamento de fluxos, como as ferramentas proprietárias de [4, 5]. O software *flow-tools* [6], 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 [7]. 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 [8] 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.

Para este artigo são propostas melhorias no modelo de [8] 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.

II. CONCEITOS GERAIS

A. Fluxos de redes

A *Cisco Systems* define um fluxo de rede como *uma seqüência unidirecional de pacotes entre hosts 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 v5*, 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 dois *hosts*, 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.

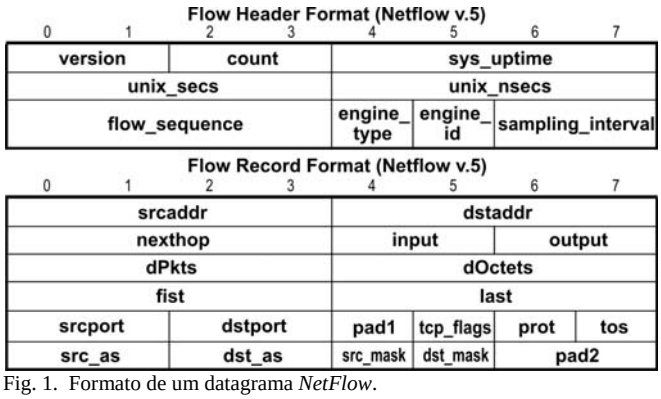

Fig. 1. Formato de um datagrama *NetFlow*.

B. Banco de dados e o modelo relacional

O Banco de dados é definido como uma coleção de dados relacionados [9]; os "dados" são fatos que podem ser gravados e que possuem um significado implícito. Um Sistema Gerenciador de Banco de Dados (SGBD) é uma coleção de programas que permite aos usuários ter um banco de dados. Assim o SGBD é um sistema de software que visa facilitar a definição, construção, manipulação e compartilhamento de banco de dados entre usuários e aplicações. Com a utilização do SGBD, surgem vantagens como:

- Controle de redundância;
- Restrição de acesso não autorizado;
- Garantia de armazenamento persistente para objetos programas;
- Garantia de armazenamento de estruturas para o processamento eficiente de consultas
- Garantia de backup e restauração;
- Fornecimento de múltiplas interfaces para os usuários;
- Representação de relacionamentos complexos entre os dados;
- Restrições de integridade;
- Permissão para inferências e ações utilizando regras.

O SGBD deve oferecer linguagens apropriadas para cada categoria de usuários. As duas principais linguagens são:

- Linguagem de Definição de Dados – *Data Definition Language* (DDL): usado para especificar os esquemas conceitual e interno;
- Linguagem de Manipulação de Dados – *Data Manipulation Language* (DML): é usado pelos usuários para manipulação dos dados como recuperação, inserção, remoção e modificação.

O modelo relacional é a representação do banco de dados como uma coleção de relações [9]. Informalmente, cada relação se parece com uma tabela de valores, 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.

Um SGBD que gerencie um modelo de banco de dados relacional tem como DDL e DML a linguagem estruturada de consulta (*Structured Query Language* – SQL) , que será vista a seguir.

C. Linguagem estruturada de consulta (SQL)

A SQL é uma linguagem de banco de dados abrangente: possui comandos para definição de dados, consultas e atualizações [9]. Assim, ela possui tanto da DDL quanto a DML. Ela também possui regras para embutir os comandos SQL em linguagens de programação de âmbito geral como Java, COBOL, ou C/C++. A SQL usa os termos **tabela**, **linha** e **coluna** em vez dos termos *relação*, *tupla* e *atributo*. Os principais comandos SQL para DDL são:

- CREATE SCHEMA: Cria um esquema de banco de dados, no qual conterá as tabelas;
- CREATE TABLE: Usado para especificar uma nova tabela, dando-lhe um nome e especificando seus atributos (colunas) e restrições iniciais. Os atributos são definidos por um nome, um tipo especificando o domínio de seus valores e uma restrição, como NOT NULL (não pode ser vazio);
- DROP: Elimina elementos de esquemas como tabelas, domínios ou restrições.
- ALTER TABLE: Altera atributos, restrições ou domínios de uma tabela.
- SELECT: Realiza consulta aos dados nas tabelas.

Um dos principais comandos SQL utilizados neste trabalho é o SELECT. A sintaxe para consultas SQL através do SELECT obedece ao seguinte padrão:

SELECT <lista de atributos> FROM <lista de tabelas> [WHERE <condição>];

- <*lista de atributos*> é uma lista dos nomes dos atributos cujos valores serão recuperados pela consulta.
- <*lista de tabelas*> é uma lista dos nomes das relações necessárias para o processamento da consulta.
- <*condição*> é uma expressão condicional (booleana) que identifica as *tuplas* que serão recuperadas pela consulta.

III. O MODELO DE ARMAZENAMENTO

Esta seção descreve as principais características do modelo de armazenamento proposto neste trabalho. Como citado anteriormente, o modelo de armazenamento é baseado no modelo de banco de dados relacional utilizando tabelas para armazenar os dados, no qual 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. 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. 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. 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 divisão de período é a que melhor se adéqua a organização dos dados e ao desempenho das consultas visto que, caso o período fosse dividido por mês, as consultas ficariam inviáveis em desempenho e, caso o período fosse dividido por semana, além do desempenho das consultas não ser satisfatório essa divisão dificulta na boa organização dos nomes das tabelas, pois há semanas no qual existe a passagem de um mês a outro.
- Tabelas adicionais:
  - Tabela de fluxos atuais: A criação de uma tabela contendo os fluxos dos últimos 30 minutos provê maior acessibilidade aos dados e geração de estatísticas. Esta tabela permitirá consultas que detectem eventos recentes na rede (como ataques a determinados *hosts*) com alto desempenho visto seu menor número de *tuplas*. Um fluxo *Netflow* chega ao coletor de forma desordenada (ordem cronológica) devida às regras de exportação e permanece no dispositivo de rede por no máximo 30 minutos antes de ser exportado [2]; assim a tabela de 30 minutos auxilia na ordenação dos dados, pois os fluxos mais antigos que este período são 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 dois *hosts*. Uma tabela denominada “input” registra os fluxos referentes ao tráfego de *hosts* externos para *hosts* internos; uma tabela denominada “output” registra os fluxos de modo inverso. Estas tabelas são úteis para cruzar informações e reconstituir conexões/sessões entre dois *hosts* 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 é construída através do produto cartesiano entre as tabelas “input” e “output”.

- Sistema Gerenciador de Banco de Dados (SGBD): A escolha do SGBD [9] a ser utilizado segue os seguintes critérios:
  - Suporte a SQL (Struct Query Language);
  - A inserção de *tuplas* no banco deve ser mais rápida que a velocidade de geração dos fluxos pelo roteador/switch. Em [8] recomenda-se utilizar um SGBD que possa inserir de forma três vezes mais rápida que a geração de fluxos em tempo real, isto porque é 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 ou restringe 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 são abordados neste artigo.

IV. A IMPLEMENTAÇÃO DO MODELO

As características principais do modelo foram estabelecidas baseadas em uma seqüência de testes realizados em busca da melhor solução para armazenamento dos fluxos. Esta seção descreve a implementação realizada e algumas considerações que levaram ao modelo proposto. Utilizou-se a versão 5 do protocolo *NetFlow* devido as restrições do ambiente de teste impostas.

A. Formato das tabelas

A criação das tabelas foi inicialmente baseada em [8]. A tabela proposta por [8] é descrita a seguir:

TABELA I
FORMATO DA TABELA PROPOSTO POR [8]

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 I, 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 [8] 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 no banco de dados serão criadas conforme Tabela II.

TABELA II
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 à proposta de [8].

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.

B. 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 resulta do produto cartesiano *Input* x *Output*, reconstituindo as conexões/sessões de modo bidirecional.

As tabelas *input* e *output* separam os fluxos do ambiente de rede monitorado com a Internet. É totalmente plausível separar fluxos cuja origem ou destino estejam dentro do próprio ambiente. No entanto, inicialmente estes fluxos não foram analisados, dando-se maior ênfase ao tráfego com a Internet. A metodologia para separação destes fluxos internos é a mesma utilizada para construir *input* e *output*, apenas mudando o valor dos campos *input* e *output* para separar os fluxos.

C. 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 [10]. 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*).

D. Sistema Gerenciador de Banco de Dados (SGBD)

Dentre os vários Sistemas Gerenciadores de Banco de Dados existentes, os não-comercais mais utilizados são MySQL [11] e Oracle [12] (versão *Free*). O trabalho de [8] descreve várias vantagens do MySQL em relação ao Oracle. Seguindo as descrições feitas por [8], 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 (notação decimal com pontos), 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*.

E. Ambiente de teste

O ambiente de testes foi montado em uma universidade e conta com mais de 1000 dispositivos de rede incluindo *hosts*, 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 80 GB ATA-100, dedicado a coleta, armazenamento e análise dos fluxos no banco (vide Figura 2).

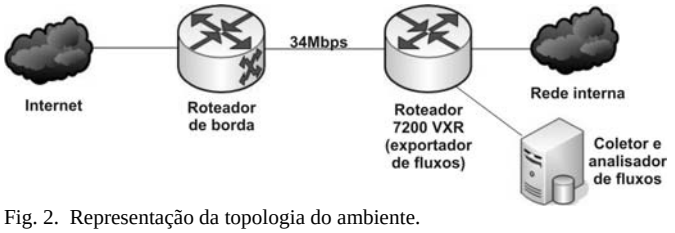

Fig. 2. Representação da topologia do ambiente.

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.

V. Consultas e Resultados

Esta seção descreve as consultas aos dados que resultam em análises de tráfego e de segurança, resultando em dados periciais. É importante observar que dados como IP de origem e destino nas consultas foram sanitizados, a fim de preservar a privacidade dos *hosts* e usuários.

A. Medição de tráfego

As consultas *i,ii* e *iii* 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    |
+----------+
| 10756345 |
+----------+
1 row in set (0.03 sec)

+---------+
| Input   |
+---------+
| 5215475 |
+---------+
1 row in set (21.89 sec)

+---------+
| Output  |
+---------+
| 5227989 |
+---------+
1 row in set (4.67 sec)
```

Fig. 3. Resultado 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. Nota-se que o valor de *input* e *output* utilizado é 28, isto porque este é o identificador da interface do roteador conectada a Internet.

As consultas *iv* e *v* 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)
```

Fig. 4. Resultado das consultas *iv* e *v*.

A consulta *vi* conta os fluxos por minuto (no período de um dia) e realiza uma média de fluxos/segundo, resultando em dados estatísticos (vide Figura 5).

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)
```

Fig. 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 [3] que utiliza fluxos de rede *NetFlow* e redes neurais para detectar anomalias de tráfego na rede.

A consulta *vii* retorna os hosts que mais receberam tráfego (em bytes) nos últimos 5 minutos (vide Figura 7). Realizando a consulta na tabela *last30minutes*, a resposta é obtida em menos de um segundo.

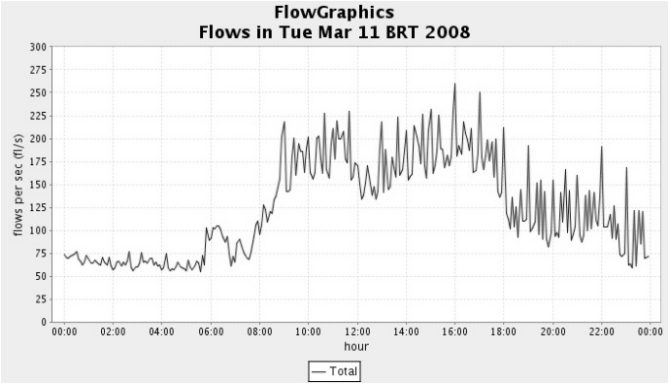

Fig. 6. Gráfico gerado por aplicação JAVA utilizando resultados de *vi*.

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)
```

Fig. 7. Resultado da consulta *vii*.

A consulta *viii* retorna os hosts que mais geraram fluxos de saída nos últimos 5 minutos (vide Figura 8). Esta consulta permite identificar *hosts* 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* [13], prospecção na rede (*portscan*) [14], *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;

B. Detecção de eventos de segurança

As consultas descritas nesta seção identificam eventos relacionados a conexões/sessões entre hosts, permitindo o levantamento de evidências sobre eventos de segurança. Os resultados destas consultas poderão auxiliar perícias forenses nos quais os fatos investigados envolvem *hosts* de uma ou mais redes, identificando os envolvidos, horários de acesso, protocolos utilizados, entre outros.

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)

Fig. 8. Resultado da consulta *viii*.

A consulta *ix* identifica a conversação entre dois *hosts* (vide Figura 9), exibindo o horário do início da conexão, a quantidade de bytes e pacotes enviados e o protocolo utilizado (TCP = 6). Nota-se que há dois fluxos resultantes nesta consulta, um indicando as informações do tráfego de ida e outra indicando as informações do tráfego de volta.

ix. SELECT inet_ntoa(srcaddr) AS Source, inet_ntoa(dstaddr) AS Destination, srcport, dstport ,first, sum(dOctets) as TotalBytes, sum(dPkts) as TotalPackets, prot as Protocol FROM afm20080506 WHERE (srcaddr = inet_aton('192.168.216.129') and dstaddr = inet_aton('192.168.202.9')) or (srcaddr = inet_aton('192.168.202.9') and dstaddr = inet_aton('192.168.216.129')) GROUP BY srcaddr,dstaddr,srcport,dstport order by first;

Source	Destination	srcport	dstport	first	TotalBytes	TotalPackets	Protocol
192.168.216.129	192.168.202.9	54616	22	2008-05-06 17:26:57	429323	5201	6
192.168.202.9	192.168.216.129	22	54616	2008-05-06 17:26:57	634087	4044	6

2 rows in set (20.73 sec)

Fig. 9. Resultado da consulta *ix*.

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)

Fig. 10. Tabela *Connections_Sessions* obtida após a execução da inserção *x*.

A inserção *x* 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 10. Como visto na Figura 10, 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.

x. 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;

A consulta *xi* identifica hosts externos realizando prospecção (*scan*) na rede interna (vide Figura 11). Para identificar esse tipo de evento é necessário filtrar padrões do mesmo, como a *flag* SYN do TCP aliada a curta duração da conexão, identificados nos campos *tcp_flags*, *first* e *last* do *NetFlow*.

xi. 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;

A consulta *xii* identifica *hosts* externos realizando ataque de dicionário no serviço SSH dos *hosts* da rede interna (vide Figura 12). 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 selecionados apenas os *hosts* que efetuaram mais de 100 tentativas em um mesmo *host* de destino.

xii. 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	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)

Fig. 11. Resultado da consulta *xi*.

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)

Fig. 12. Resultado da consulta *xii*.

A consulta *xiii* identifica possíveis *hosts* da rede interna utilizando aplicativos *file sharing* (vide Figura 13). 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.

xiii. 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;

C. 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 III 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 os 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 *hosts* realizando *scan* ou *file sharing*.

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)

Fig. 13. Resultado da consulta *xiii*.

TABELA III
COMPARAÇÃO DE TEMPO EM SEGUNDOS PARA REALIZAR CONSULTA AOS FLUXOS

Consulta	**Banco**	***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 hosts com maior número de octetos recebidos (24 horas).	30.76s	59.43s

A Tabela IV 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 IV
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

VI. CONCLUSÃO

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, dados fornecidos pelo protocolo *NetFlow* armazenados em um banco de dados relacional facilitam o gerenciamento dos dados, além de prover 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. Esses eventos detectados com precisão servirão como dados periciais a análises forenses auxiliando profissionais desta área.

As consultas da seção V subseção A 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 V subseção B manipulam os dados dos fluxos para detectar eventos de segurança na rede, como a detecção de *hosts* realizando prospecção na rede, ataques de dicionário no serviço SSH ou utilizando programas *file sharing*, além de obter diversas informações relacionadas a uma conexão/sessão entre dois *hosts*. Todas essas consultas permitem ao analista de segurança uma análise pericial apurada do tráfego da rede sob sua administração, auxiliando em eventuais investigações envolvendo crimes cibernéticos.

Uma questão que fica pendente após a demonstração das consultas é a de como realizar as mesmas consultas para intervalos de vários dias, já que cada tabela contém um único dia de fluxos. A resposta vem com a utilização do recurso *UNION* [11] presente na linguagem SQL. Este recurso possibilita a união de várias consultas em um único comando. Assim para cada tabela diária constrói-se uma consulta e utiliza-se o *UNION* para unir os resultados.

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.

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, possibilitando adaptar o sistema para a análise de *links* de grande porte. 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

[1] J. Quittek, T. Zseby, B. Claise, and S. Zander, (2004) "RFC 3917: Requirements for IP Flow Information Export: IPFIX," [Online]. Available: http://www.ietf.org/rfc/rfc3917.txt

[2] B. Claise, (2004) "RFC 3954: Cisco Systems NetFlow Services Export Version 9," [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[3] A. M. Cansian and J. L. Corrêa, "Detecção de ataques de negativa de serviço por meio de fluxos de dados e sistemas inteligentes," *VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, vol. 7, pp. 125-141, 2007.

[4] AdventNet, (2007) "ManageEngine NetFlow Analyzer," [Online]. Available: http://origin.manageengine.adventnet.com/products/netflow/index.html

[5] F. Networks, (2008) "NetFlow Tracker," [Online]. Available: http://www.flukenetworks.com/fnet/en-us/products/NetFlow+Tracker/

[6] M. Fullmer, (2007) "Tool set for working with NetFlow data," [Online]. Available: http://www.splintered.net/sw/flow-tools/docs/flow-tools.html

[7] P. Dave, "FlowScan: A Network Traffic Flow Reporting and Visualization Tool," *Proceedings of the 14th USENIX conference on System administration*, pp. 305-318, 2000.

[8] N. Bill, "Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics," *Proceedings of the 14th USENIX conference on System administration*, pp. 285-290, 2000.

[9] R. E. ELMASRI and S. NAVATHE, *Sistemas de Banco de Dados* vol. 4: Addison-Wesley, 2005.

[10] C. Systems, (2004) "NetFlow Packet version 5 (V5)," [Online]. Available: http://netflow.caligare.com/netflow_v5.htm

[11] MySQL, (2008) "MySQL 5.1 Reference Manual," [Online]. Available: http://dev.mysql.com/doc/refman/5.1/en/index.html

[12] Oracle, (2008) "Oracle Database Documentation," [Online]. Available: http://www.oracle.com/technology/documentation/database.html

[13] Cert.br, (2006) "Cartilha de Segurança para Internet - Parte VIII: Códigos maliciosos (Malware)," [Online]. Available: http://cartilha.cert.br/malware/

[14] Fyodor, (1997) "The Art of Port Scanning," [Online]. Available: http://www.insecure.org/nmap/nmap_doc.html