

# SniffStat2DB – Ferramenta para Coleta e Armazenamento dos Dados do Tráfego de uma Rede\*

Adler H. Schmidt<sup>1</sup>, Tiago Perlin<sup>1</sup>, Raul C. Nunes<sup>1</sup>

<sup>1</sup>Instituto de Informática – Universidade Federal de Santa Maria (UFSM)  
Santa Maria – RS – Brazil

{adlerhs, tiagoperlin}@gmail.com, ceretta@inf.ufsm.br

**Abstract.** *Anomalies based intrusion detection systems make an analysis of the behavior of a network an estimate the probability of a given situation be an attack. So that these systems can work live it is necessary a tool that makes the gathering and quantification of the data that is being transmitted on this network. This article will describe a work that has been done to build a network data collector that has as a goal to facilitate at best the using of these data by the described systems. Doing so, that the access to these network counters, that represents the situation of the network along time, be easy and fast.*

**Resumo.** *Sistemas de detecção de intrusão por anomalias realizam uma análise do comportamento de uma rede e procuram avaliar se a situação atual pode ser um ataque. Para que esses sistemas possam operar em tempo real é necessária uma ferramenta que realize uma coleta e quantificação dos dados que estão transitando na rede. Esse artigo descreve a construção de um coletor de tráfego de rede que visa facilitar a utilização dos dados pelos sistemas de detecção de intrusão, fazendo com que o acesso aos quantificadores de estado da rede seja prático e rápido.*

## 1. Introdução

Com a proliferação do uso de redes de computadores os mais diversos tipos de serviços foram incorporados a essa forma de acesso, devido à praticidade que propicia. Porém usuários mal-intencionados sempre surgem com o intuito de causar um comportamento irregular a esse meio de comunicação. Na tentativa de evitar que ataques aconteçam, várias medidas foram e vem sendo tomadas. Uma delas que podemos destacar é o uso de sistemas para a detecção de intrusão baseados em anomalias. Esses sistemas têm como objetivo prever um ataque através de uma comparação entre o comportamento atual de uma rede de computadores e o comportamento passado, em momentos que se sabia não estarem ocorrendo ataques [Barford et al. 2002]. Essa comparação se dá através de dados quantificados sobre todo tráfego que está ocorrendo. Porém, essa quantificação em si foge do escopo de um sistema de detecção de intrusão. Ela é realizada por um coletor de tráfego auxiliar, ferramenta que se responsabiliza pela escuta do que está acontecendo na rede e pelo armazenamento desses dados. Coletores de tráfego também são referenciados como *sniffers* [Tanase 2002]. Junto com a dependência entre esses

\* Trabalho parcialmente apoiado pela FAPERGS (BIC Proc. 08514069) e pelo Convênio INPE - UFSM

dois sistemas vêm vários problemas de compatibilidade. Nem sempre é possível usar o mesmo *sniffer* para sistemas que realizam de forma específica a detecção de intrusão. Cada uma das partes precisa estar em harmonia.

O sniffer Internet Analysis System - IAS [Pohlmann and Proest 2006] foi utilizada como ferramenta para coleta, em tempo real, dos contadores de estado da rede nos sistemas de detecção de intrusão por anomalias DIBSeT [Lunardi et al. 2008] e DIBSeT-W [Dalmazo et al. 2008]. Porém, esse sniffer foi criado com objetivo específico: armazenar o máximo de informações possíveis sobre o tráfego de uma rede, o que resultava em vários empecilhos para utilizá-lo. Os dois principais eram o baixo desempenho para acesso aos dados e a necessidade de realizar um trabalho adicional após o acesso para quantificar todos dados, o que agregava um custo à operação.

Este trabalho apresenta uma solução otimizada para realização de *sniffing* para quantificação dos pacotes, objetivando um acesso o mais eficaz (configurável, específico e veloz). A ferramenta resultante dessa solução tem como foco principal prover uma base para sistemas de detecção de intrusão por anomalias que utilizam séries temporais ou qualquer outro cliente que necessite somente de quantificadores e não de todos os dados possíveis de cada pacote, como nos sistemas análise de perigo na internet [Pohlmann and Proest 2006].

O trabalho está organizado como segue: a seção 2 apresenta os SDI DIBSeT [Lunardi et al. 2008] e o DIBSeT-W [Dalmazo et al. 2008], seguida da ilustração do problema que ambos vivenciavam. Na seção 3 é feito uma apresentação de trabalhos que sugeriram soluções similares para problemas bem parecidos. Na seção 4 é apresentada a solução que esse trabalho propõe. Finalmente a seção 5 apresenta os resultados obtidos e a seção 6 conclui o trabalho.

## **2. Trabalhos Anteriores**

Para facilitar o entendimento, essa seção será subdividida novamente em outras duas que irão fazer uma breve descrição do trabalho feito no DIBSeT [Lunardi et al. 2008] na seção 2.1 e as modificações que resultaram no DIBSeT-W [Dalmazo et al. 2008] na seção 2.2.

### **2.1. DIBSeT**

O DIBSeT [Lunardi et al. 2008] é um sistema de detecção de intrusão que visa encontrar anomalias nos contadores de tráfego da rede utilizando séries temporais do modelo ARIMA e o preditor de passos em séries temporais desenvolvido em [Nunes 2003]. Com ele, limites dinâmicos são estabelecidos para estipular a probabilidade de que uma anomalia se trata de um ataque quando os contadores ultrapassarem os mesmos.

O DIBSeT possuía como opção o uso do IAS [Pohlmann and Proest 2006] como ferramenta auxiliar para *sniffing*. Porém, essa utilização recaiu sobre o problema de compatibilidade entre ferramentas para *sniffing* e os sistemas de detecção de intrusão que utilizam os dados gerados. Enquanto o IAS armazena uma série muito grande de dados, tentando capturar dados para especificar em detalhes como o tráfego da rede está ocorrendo, o DIBSeT necessitava somente dos contadores de determinados tipos de pacotes em certos intervalos de tempo para construir as suas séries temporais e analisar

ataques. Dessa maneira o custo para trabalhar com os dados fornecidos pelo IAS tornou-se muito alto, o que inviabilizou o seu uso.

## 2.2. DIBSeT-W

O DIBSeT-W [Dalmazo et al. 2008] teve como foco a melhoria da qualidade nos resultados do DIBSeT, através da redução dos falsos positivos, previsões de que estava ocorrendo um ataque na rede quando na realidade não estava. Com o uso de *wavelets* [Mallat 1989] foi possível realizar essa filtragem de respostas indesejadas. Porém, vale ressaltar que as séries temporais continuaram a ser utilizadas e a cooperação entre IAS e DIBSeT mantinha-se um problema.

Sendo assim, a utilização de um *sniffer* não direcionado especificamente para o uso de construção de séries temporais se apresentava como custosa demais. A criação de um coletor e armazenador de dados do tráfego da rede que resolvesse especificamente esse problema viu-se como uma boa contribuição a ser realizada.

## 3. Trabalhos Relacionados

O mecanismo de funcionamento dos *sniffers* dividem-se em dois passos a serem abordados: a coleta dos dados e o armazenamento dos mesmos. O armazenamento pode ser de várias formas, variando de acordo com o objetivo que se quer atender, porém pela parte da coleta as soluções geralmente se dividem novamente em duas abordagens:

- Ferramentas para *sniffing* de placas de rede como o TCPDUMP [Tcpdump 2009], o WIRESHARK [Wireshark 2009] ou até mesmo de implementações próprias utilizando a biblioteca LIBPCAP [Libpcap 2009]. Nesse caso geralmente utiliza-se uma máquina como gateway que fica entre a rede externa e a interna;
- Utilização de ferramentas que visionam facilitar esse trabalho, como o *NetFlow* [Claise 2004], que é baseado no protocolo IPFIX [Quittek et al. 2004]. Os dados IPFIX de cada pacote que está transitando por um dispositivo que suporta o *NetFlow* são capturados e então enviados para um cliente dedicado que está a processá-los.

Baseado na segunda abordagem, [Correa et al. 2008] propõe um modelo de armazenamento de fluxos de rede para análise de tráfego e de segurança, onde os dados são coletados em um roteador compatível que, por sua vez, envia esses dados ao responsável por tratá-los. Outras duas idéias interessantes apresentadas em [Correa et al. 2008] são a da criação de uma tabela de acesso rápido, tabela que guardará somente os dados dos últimos trinta minutos para um acesso mais rápido aos dados coletados, e de uma tabela com armazenamento diário das informações.

Para facilitar o entendimento podemos dividir a solução proposta por [Correa et al. 2008] nos três seguintes passos:

- Recebimento dos dados de tráfego com o protocolo *NetFlow* [Claise 2004];
- Inserção dos mesmos em uma tabela de acesso rápido;
- Manutenção dessa tabela de acesso rápido e transferência dos dados mais antigos para uma tabela de armazenamento.

Certamente outras soluções para coleta e armazenamento existem e pode-se citar as ferramentas pagas [Adventnet 2009] e [Networks 2009] que também utilizam o *NetFlow*.

#### **4. SniffStat2DB – Ferramenta de Coleta e Armazenamento de Dados de Rede**

Para melhor entendimento do que será proposto ocorrerá uma divisão dessa seção em quatro subseções: a capacidade de mudança de comportamento do programa de acordo com as necessidades do usuário (4.1), a coleta dos dados (4.2), o armazenamento deles (4.3), e uma última seção que irá apresentar detalhes que possibilitam a execução em paralelo (4.4).

##### **4.1. Configurações Pré-Execução**

Um dos principais objetivos do SniffStat2DB é fornecer uma solução adaptável de acordo com as necessidades de cada usuário, neste trabalho os SDI DIBSeT e DIBSeT-W. Neste sentido, ao invés de coletar dados sobre todos os pacotes de rede, por host e por porta, tal como o IAS faz, o SniffStat2DB utiliza um arquivo de configuração que possibilita ao usuário determinar o funcionamento da ferramenta, tal como quais os pacotes de rede devem ser filtrados. Deste modo, o usuário pode alterar o comportamento do SniffStat2DB ao alterar o conteúdo do arquivo de configuração, que é um arquivo (XML) [XML 2009]. Uma regra para a sintaxe desses arquivos XML foi criada e as seguintes informações devem estar contidas nesse arquivo:

- Para captura dos dados: porta que se deseja filtrar, intervalo de análise entre cada contagem e quantidade de intervalos (ou valor negativo para execução indefinida);
- Para armazenamento na base de dados: identificador do SGBD utilizado, nome do *host*, porta para acesso, endereço da base, nome de usuário, senha do usuário, caso deseja-se utilizar a tabela de acesso rápido, o identificador dela, caso deseja-se utilizar a tabela de arquivamento e o seu identificador;
- Filtros de captura: representação de cada uma das possibilidades de filtros (os seis citados) de forma binária que indica a utilização do filtro;
- Para envio dos dados por Sockets: o desejo de utilizá-los e a porta em que será criado o servidor para envio.

##### **4.2. Coleta de Dados**

Diferentemente da solução proposta por [Correa et al. 2008], a qual utilizou-se da ferramenta proprietária *NetFlow*, o SniffStat2DB utiliza um computador *gateway* e faz uso da biblioteca [Libpcap 2009]. As duas principais razões para essa escolha foram: a facilidade para utilizar em uma rede e os filtros de captura, responsáveis por realizar a separação de quais pacotes, dos que estão transitando, e *display*, que irão realizar um trabalho adicional em cima dessa informação para ser apresentada ao cliente, poderem ser utilizados diretamente na coleta dos dados.

Sem a exigência do *NetFlow*, basta implantar o SniffStat2DB em uma máquina que seja um *gateway* da rede e suporte a LIBPCAP. Por outro lado, com filtros de captura separados dos de *display*, pode-se melhorar o desempenho aplicando os filtros de *display* apenas no resultado dos filtros de captura, ao invés de ter que aguardar a captura

de todos os dados de tráfego e aplicar um filtro de *display* para seleccionar o que o usuário deseja. Comparando a LIBPCAP com as outras ferramentas para captura direta de uma placa de rede, tal como a TCPDUMP [Tcpdump 2009], que não fornece filtros de display, e a WIRESHARK [Wireshark 2009], que aplica os filtros de display somente após os filtros de captura, a LIBPCAP apresenta melhor desempenho. Um dos motivos é que ambas ferramentas (TCPDUMP e WIRESHARK) são implementações em cima da LIBPCAP [Libpcap 2009]. Como o SniffStat2DB foi implementado em Java, utilizou-se apenas um invólucro da biblioteca LIBPCAP conhecido como JNetPcap [JNetPcap 2009].

Nas próximas duas subsecções serão explicadas as funções de cada um dos filtros aplicados no SniffStat2DB:

#### **4.2.1. Filtros de Captura**

Na primeira versão do SniffStat2DB os tipos de pacotes que são filtrados na captura dos dados foram determinados pelas necessidades dos detectores de intrusão descritos na seção 2, os quais têm como objetivo principal combater ataques de negação de serviços (DoS - *Denial Of Service*). Deste modo, focou-se nos tipos de pacotes envolvidos nos principais ataques DoS: pacotes do protocolo TCP, UDP, ICMP e também dentre os pacotes TCP aqueles que possuem as *flags* ACK, FIN ou SYN ativada.

#### **4.2.2. Filtros de Display**

Como o objetivo desse trabalho é armazenar somente os contadores de certos pacotes que estão trafegando na rede, os filtros de *display* realizam a contagem de todos os pacotes fornecidos pelos filtros de captura, incluindo as *flags* TCP, para então realizar o armazenamento apenas das informações relevantes.

### **4.3. Armazenamento dos Dados**

Realizada a captura dos dados, o armazenamento no SniffStat2DB possibilita duas opções: o repasse da responsabilidade de armazenamento ao cliente, fazendo o envio dos dados coletados diretamente ao cliente através de *sockets* TCP; e o armazenamento em um banco de dados relacional [Elmasri and Navathe 2005]. As próximas duas subsecções irão detalhar cada uma destas opções.

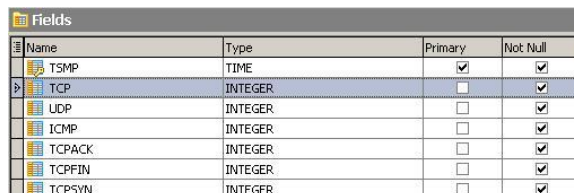
#### **4.3.1. Envio dos Dados por Sockets**

O envio direto tem o propósito de ser a maneira mais rápida e de baixo custo para o repasse das informações capturadas. A sua principal vantagem é a conexão direta com o sistema detecção de intrusão, sem qualquer outro meio que possa agregar algum custo. Porém a função de armazenar os dados passa a ser do detector de intrusão e não mais do SniffStat2DB.

#### **4.3.2. Modelo de Dados Relacional**

Um conjunto de relações segundo [Elmasri and Navathe 2005] são tabelas que possuem um conjunto de tuplas, linhas que representam um conjunto de valores, e atributos, que

são as colunas com um significado para cada um dos valores da tupla. No SniffStat2DB as colunas representam os contadores de cada um dos tipos de pacotes escolhidos e o momento em que iniciou a contagem de cada linha. Deste modo, as tuplas serão representações dos contadores dos dados capturados em certo intervalo de tempo. O identificador do início desse intervalo de tempo é a chave principal de cada tabela. Um exemplo de cabeçalho de uma tabela do SniffStat2DB pode ser visto na figura 3.



Name	Type	Primary	Not Null
TSMP	TIME	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
TCP	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>
UDP	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ICMP	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>
TCPACK	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>
TCPFIN	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>
TCPSYN	INTEGER	<input type="checkbox"/>	<input checked="" type="checkbox"/>

**Figura 3. Cabeçalho de uma tabela do SniffStat2DB em uma execução que esteja filtrando todos os pacotes**

De maneira similar ao realizado em [Correa et al. 2008], para um melhor desempenho no SniffStat2DB são criadas duas tabelas: uma para acesso rápido com os últimos trinta minutos e outra para arquivamento. A única diferença em relação a [Correa et al. 2008] é a de que as tabelas não possuem uma coluna adicional referente à porta que os contadores representam, pois cada tabela representa as contagens de uma porta específica ou de todas as portas. Essa é uma medida adotada para melhoria no acesso das informações guardadas, pois assim ao invés de se ter tabelas de grande proporção tem-se várias tabelas referentes à cada porta de interesse e registros de quando cada captura foi realizada.

A escolha do Sistema Gerenciador de Banco de Dados (SGBD) a ser utilizado também é configurável de acordo com a preferência do usuário dentre as seguintes opções: Firebird [Firebird 2009], MySQL [MySQL 2009] e PostgreSQL [PostgreSQL 2009]. Esses três SGBDs foram escolhidos devido ao fato de serem os mais populares das possibilidades gratuitas.

#### 4.4. Exploração de Paralelismo

O SniffStat2DB explora o paralelismo fazendo uso de threads. Há a thread principal é responsável pela chamada dos métodos da LIBPCAP [Libpcap 2009] e pelo armazenamento em uma estrutura de dados temporária, outra para envio dos dados por sockets, outra para controlar a inserção de dados na tabela de acesso rápido e outra para escrita na tabela de arquivamento. Essas opções podem operar ao mesmo tempo, o que pode resultar em um total de quatro threads de execução. Num gateway multicore esta forma de funcionamento potencializa o paralelismo e o desempenho do SniffStat2DB.

### 5. Resultados

Os resultados obtidos com o SniffStat2DB estão diretamente relacionados com dois fatores: o custo para repasse dos dados capturados e o custo para armazenamento nas tabelas de dados. Quanto ao custo de repasse, temos o acesso por comunicação de *sockets* e um acesso à base de dados.

Uma troca de mensagens por *sockets* entre a mesma máquina ocorreu um atraso médio de 0,015 segundos no envio do primeiro intervalo de captura. Porém, do segundo

em diante esse atraso virou insignificante (menor do que 1 milésimo de segundo), com o SDI recebendo os dados quase que instantaneamente. Isso se deu graças à arquitetura com execução em paralelo.

Quanto ao acesso ao banco de dados foi realizada uma comparação com o IAS utilizando-se o MySQL [MySQL 2009], que é o único suportado pelo IAS. O IAS realiza o armazenamento dos dados do tráfego em três tabelas e em intervalos de cinco minutos, da mesma maneira foi construída uma tabela equivalente com o SniffStat2DB em intervalos de 5 minutos. A partir disso foram realizadas dez consultas para obter-se os indicadores do tráfego nos intervalos de tempo. Enquanto o IAS retornava somente um tipo de pacote por consulta, fazendo necessário seis consultas para obter-se todos os contadores, o SniffStat2DB retornava todos em uma só consulta. O tempo levado para conexão ao banco de dados foi ignorado, somente contabilizou-se o tempo para realizar a consulta e obter os contadores. Na tabela 1 estão os dados comparativos entre o IAS e o SniffStat2DB.

**Tabela 1. Medidas de custo (em segundos) para acesso aos contadores do tráfego**

Tentativa	IAS	SniffStat2DB
1	8.882	0.125
2	8.902	0.109
3	8.932	0.125
4	8.910	0.125
5	8.929	0.109
6	8.896	0.125
7	8.906	0.110
8	8.900	0.125
9	8.929	0.110
10	8.920	0.109
Média	8.9106	0.1172

## 6. Conclusão

A iniciativa de realizar um sistema de captura e armazenamento de dados do tráfego de uma rede específico para clientes que constroem séries temporais, ou quaisquer gráficos de valores quantificados, possibilitou otimizar a coleta de dados dos detectores de intrusão DIBSeT e DIBSeT-W, pois com o SniffStat2DB tanto o tamanho das tabelas geradas quanto a velocidade de acesso aos dados diminuíram drasticamente.

## Referências

- Adventnet (2009). Manage engine netflow analyzer. Disponível em: <http://www.manageengine.com/products/netflow/index.html> Acesso em: Junho de 2009.
- Barford, P., Kline, J., Plonka, D., and Ron, A. (2002). A signal analysis of network traffic anomalies. Proceedings of ACM SIGCOMM Internet Measurement Workshop 2002.

- Claise, B. (2004). Rfc3954 - cisco systems netflow services export version 9. Disponível em: <http://www.ietf.org/rfc/rfc3954.txt> Acesso em: Junho de 2009.
- Correa, J. L., Proto, A., and Cansian, A. M. (2008). Modelo de armazenamento de fluxos de rede para análises de tráfego e de segurança. VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais.
- Dalmazo, B. L., Vogt, F., Perlin, T., and Nunes, R. C. (2008). Detecção de intrusão baseado em séries temporais. Anais do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais.
- Elmasri, R. E. and Navathe, S. (2005). “Sistemas de Banco de Dados”, Addison-Wesley, ISBN 8588639173.
- Firebird (2009). Disponível em: <http://www.firebirdsql.org/> Acesso em: Junho de 2009.
- JNetPcap (2009). Disponível em: <http://jnetpcap.com/> Acesso em: Julho de 2009.
- Libpcap (2009). Disponível em: <http://www.tcpdump.org/> Acesso em: Junho de 2009
- Lunardi, R., Dalmazo, B. L., Érico Amaral, and Nunes, R. C. (2008). Dibset: um detector de intrusão por anomalias baseado em séries temporais. Anais do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais.
- Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11:674– 693.
- MySQL (2009). Disponível em: <http://www.mysql.com/> Acesso em: Julho de 2009.
- Networks, F. (2009). Netflow tracker. Disponível em: <http://www.flukenetworks.com/fnet/en-us/products/NetFlow+Tracker/> Acesso em: Junho de 2009.
- Nunes, R. C. (2003). Adaptação dinâmica do timeout de detectores de defeitos através do uso de séries temporais. Programa de Pós-Graduação em Computação: Tese de Doutorado Universidade Federal do Rio Grande do Sul, Porto Alegre.
- Pohlmann, N. and Proest, M. (2006). Internet Early Warning System: The Global View.
- PostgreSQL (2009). Disponível em: <http://www.postgresql.org/> Acesso em: Julho de 2009.
- Quittek, J., Zseby, T., Claise, B., and Zander, S. (2004). Rfc3917 - requirements for ip flow information export: Ipflix. Disponível em: <http://www.ietf.org/rfc/rfc3917.txt> Acesso em: Junho de 2009.
- Tanase, Matthew (2002). Disponível em: <http://www.securityfocus.com/infocus/1549> Acesso em: Agosto de 2009.
- Tcpdump (2009). Tcpdump. Disponível em: <http://www.tcpdump.org/> Acesso em: Junho de 2009.
- XML (2009). Disponível em: <http://www.w3.org/XML/> Acesso em: Junho de 2009.
- Wireshark (2009). Disponível em: <http://www.wireshark.org> Acesso em: Junho de 2009.