Aplicando a Frequência de Episódios na Correlação de Alertas

Leonardo Vilaça Silva[1], Eduardo Luzeiro Feitosa[12], Djamel Sadok[1], Judith Kelner[1]

[1]Centro de Informática
Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – Cidade Universitária - Recife - PE

[2]Departamento de Ciência da Computação
Universidade Federal do Amazonas (UFAM)
CEP 69077-000 Campus Universitário - Manaus - AM

{lhvs, elf, jamel, jk}@cin.ufpe.br

Abstract. *In the current security situation of computer networks, traffic anomalies generated by many different reasons have affected an increasing portion of the Internet. This paper presents a tool for correlation and prediction of alerts based on frequent episodes technique to detect anomalous traffic. For validation, tests will be done in a controlled environment using DARPA 2000 dataset and real traffic.*

Resumo. *Na atual situação de segurança das redes de computadores, anomalias de tráfego geradas pelos mais diversos motivos têm afetado uma porção cada vez maior da Internet. Este artigo apresenta uma ferramenta de correlação e predição de alertas baseada na técnica de episódios frequentes para detecção de tráfego anômalo. Para validação, serão realizados testes usando a base DARPA 2000 e tráfego real.*

1. Introdução

Nos últimos anos, os ataques a protocolos e serviços, a proliferação de vírus e *worms*, e outras atividades maliciosas tem se tornando mais comuns e de difícil detecção, causando anomalias em softwares, sistemas e redes vulneráveis e resultando em desperdício de recursos, prejuízos financeiros, degradação do desempenho e queda na confiabilidade.

Embora soluções de filtragem e software "anti alguma coisa" sejam muito usadas, sistemas de detecção e prevenção como IDS (*Intrusion Detection Systems*), IPS (*Intrusion Prevention Systems*) e ADS (*Anomaly Detection Systems*) são as mais efetivas. Contudo, elas enfrentam problemas como: (i) a geração de grandes quantidades de alertas; (ii) a não detecção de ataques novos ou variações de ataques conhecidos; e (iii) visão focada em ataques e anomalias de baixo nível, ignorando a lógica ou as estratégias envolvidas nos ataques [Feitosa *et al*. 2008].

É neste contexto que este trabalho propõe o uso da técnica de descoberta de episódios frequentes (do inglês *Frequent Episodes Discover* - FED) para aprimorar os resultados das detecções. Em linhas gerais, o objetivo é desenvolver um sistema de correlação, e possível predição, de alertas capaz de receber um conjunto de alertas como

entrada, processá-los e gerar regras que confirmem a identificação de ataques e anomalias, podendo, assim, permitir a predição de futuros eventos.

O restante desse artigo é organizado da seguinte forma: a seção 2 apresenta os conceitos básicos relativos à técnica de descoberta de episódios frequentes; a seção 3 apresenta os trabalhos relacionados; a seção 4 descreve a solução proposta, incluindo a arquitetura e desenvolvimento de um protótipo funcional; a seção 5 trata da análise dos resultados obtidos durante testes e experimentos; por fim, a seção 6 apresenta as conclusões finais e perspectivas de trabalhos futuros.

2. Descoberta de Episódios Frequentes

A técnica de episódios frequentes é voltada para descoberta de padrões temporais em dados sequenciais. Proposto por Mannila, Toivonen e Verk [Manilla *et al.* 1997], a técnica baseia-se em dois conceitos chave: sequência de eventos e episódio. O primeiro refere-se ao comportamento ou ações (de usuários ou sistemas) que podem ser coletados em diversos domínios. O segundo é uma coleção de eventos que ocorrem relativamente próximos uns dos outros em uma ordem parcial. De modo geral, o objetivo da FED é analisar, em uma sequência de eventos, quais episódios são frequentes.

2.1 Sequência de Eventos

Dado um conjunto *E* de tipos de eventos, um evento é representado pelo par (*A*, *t*), onde $A \in E$ é um tipo de evento e *t* é um inteiro que representa o tempo de ocorrência de um evento. Sendo assim, uma sequência de eventos **s** em *E* é um conjunto (s, T_s, T_e), onde

$$s = < (A_1, t_1), (A_2, t_2), \dots, (A_n, t_n) > \quad \textbf{(1)}$$

é uma sequência de eventos ordenada tal que $A_i \in E$ para todo $i = 1, \dots, n$, e $t_i \leq t_{i+1}$ para todo $i = 1, \dots, n-1$. Além disso, T_s e T_e são dois inteiros que representam o tempo de início e término, respectivamente, e $T_s \leq t_i < T_e$ para todo $i = 1, \dots, n$.

A figura 1 ilustra uma sequência **s** com 10 eventos no intervalo de tempo entre 59 e 83.

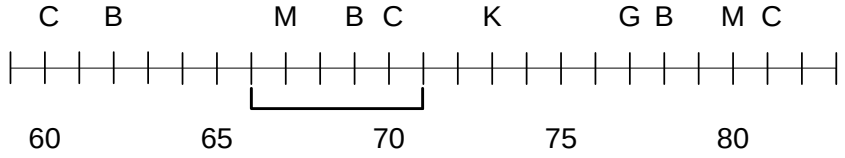


Figura 1. Representação gráfica de uma sequência de eventos s

2.1.1 Janela de Tempo

Por definição, uma **janela de tempo** é uma fatia de uma sequência de eventos. O tamanho da janela é definido pelo usuário. Na Figura 1, a janela de tempo tem tamanho 5 (entre o intervalo 66 e 71).

Formalmente, uma janela de tempo em uma sequência **s** = (s, T_s, T_e) é uma sequência de eventos **w** = (w, t_s, t_e), onde $t_s < T_e$ e $t_e > T_s$, e *w* é formado por um par (*A*, *t*) de **s** onde $t_s \leq t < t_e$. O intervalo de tempo entre t_e e t_s, $t_s - t_e$, é chamado de tamanho da janela **w** e é representada por *width* (**w**). Desta forma, dado uma sequência de evento **s** e um valor inteiro *win*, o conjunto de todas as janelas **w** de tamanho *win* na sequência **s** são denotadas por $W(s, win)$. Ainda de acordo com a definição, a primeira e a última janela em uma sequência se estendem para fora da sequência, tal que a

primeira janela contenha somente o primeiro ponto de tempo da sequência e a última janela contenha apenas o último ponto de tempo, permitindo que um evento seja observado igualitariamente em uma sequência.

2.2 Episódios

Episódios podem ser definidos como uma coleção de eventos ordenados parcialmente de acordo com sua ocorrência. Formalmente, um episódio pode ser definido como $(V, \leq, m)$, onde *V* é o conjunto de todos os nós do episódio, o símbolo ≤ é a ordem em que os eventos ocorrem no episódio, e *m* é a função que faz o mapeamento (*m*: $V \rightarrow$ *Seq*) dos nós com seus respectivos tipos de eventos em uma sequência de eventos.

2.3 Frequência

A frequência de um episódio é definida como sendo a razão entre o número de janelas em que o episódio ocorre e o número total de janelas. Isto é, dada uma sequência de eventos *s* e uma janela de tamanho *win*, a frequência de um episódio *E* em *s* é:

$$fr(E, s, win) = \frac{|\{\boldsymbol{w} \in W(\boldsymbol{s}, win) | E \ ocorre \ em \ \boldsymbol{w}\}|}{|W(\boldsymbol{s}, win)|} \quad \textbf{(2)}$$

Para determinar se um episódio *E* é ou não frequente, um **limite de frequência** (*min_fr*) é utilizado. Sendo assim, *E* é dito frequente se $fr(E, s, win) \geq min\_fr$. A representação do conjunto de episódios frequentes em relação à *s* é dada por $\mathcal{F}(s, win, \min\_fr)$. É importante ressaltar que se um episódio é frequente, então todos os seus subepisódios também serão frequentes.

2.3.1 Descoberta da frequência dos episódios

Uma vez que os episódios frequentes são conhecidos, eles podem ser utilizados na obtenção de correlações entre os eventos da sequência. Essas relações são chamadas de **regras de episódio**. Uma regra entre dois episódios *X* e *Y* é definida formalmente como $X \Rightarrow Y$, denominada R_{xy}, se *X* é subepisódio de *Y*.

Por exemplo, se os episódios $(A \rightarrow B)$ e $(A \rightarrow B \rightarrow C)$ são frequentes, com frequências f_1 e f_2 respectivamente, a regra resultante é $(A \rightarrow B) \Rightarrow (A \rightarrow B \rightarrow C)$ se a **confiança** estabelecida $(\frac{f_2}{f_1})$ ultrapassar um limite pré-definido. A confiança de uma regra representa a probabilidade condicional do episódio *Y* ocorrer por completo em uma janela, dado que o episódio *X* ocorreu na mesma.

[Mannila *et al.* 1997] propõem duas abordagens para o cálculo da frequência dos episódios: uma baseada no número de janelas e outra em ocorrências mínimas. Neste trabalho, somente a primeira será utilizada.

3. Trabalhos Relacionados

O uso de episódios frequentes na detecção de anomalias não é uma novidade. Lee, Storfo e Mok [Lee *et al.* 2000] elaboraram um arcabouço (*framework*) de mineração de padrões para realizar a detecção de intrusão, visando a criação de um sistema de auditoria para encontrar padrões no comportamento de usuários e programas. No trabalho de Qin e Hwang [Qin and Hwang 2004], foram desenvolvidos algoritmos para reduzir a quantidade de regras geradas pelo cálculo da frequência de episódios,

diminuindo o espaço de busca das anomalias em 80%. Foram realizados testes com o DARPA 1999 a fim de comprovar a eficiência do novo método.

Hwang *et al.* [Hwang *et al.* 2007] desenvolveu um sistema de detecção híbrido, combinando as vantagens da baixa taxa de falso positivo dos IDSs com a habilidade de detectar ataques desconhecidos de ADS, onde a frequência de episódios foi utilizada para criar uma associação entre as regras geradas, a fim de permitir a detecção de ataques similares. Em [Soleimani and Ghorbani 2008], os autores propuseram uma modificação da técnica de episódios frequentes, objetivando gerenciar as grandes quantidades de alertas gerados pelos IDSs. O foco dado foi na descoberta de todas as possíveis sequências de alertas e suas combinações.

Embora ainda existam vários outros trabalhos envolvendo a aplicação da técnica de episódios frequentes, a principal contribuição (e diferencial) deste trabalho reside no fato de que ao invés de usar FED na análise do tráfego e na posterior geração dos alertas, ele usa a FED para extrair eventos de segurança (alertas) gerados por múltiplos detectores. Desta forma, é possível correlacionar alertas provenientes de diferentes localizações da rede e gerados por diferentes técnicas de detecção, diminuindo assim a incerteza de um ataque ou anomalia em curso e permitindo a criação de uma "base de padrões" que pode ser usada na predição de futuros eventos similares.

4. Projeto e Implementação

A solução proposta foi projetada com o objetivo de aumentar a precisão dos alertas e prever futuros alertas de acordo com o estado da rede. Na prática representa um software capaz de efetuar a correlação de alertas e gerar regras sobre os episódios frequentes que podem ser aplicadas na contenção imediata do ataque ou anomalia e na geração de padrões que permitam a predição de futuros eventos similares.

4.1 Arquitetura

Em relação à arquitetura, a solução proposta foi projetada de forma modular, o que facilita a manutenção, diminui a repetição de código e melhora sua legibilidade. Todos os componentes foram desenvolvidos em Java (versão 1.6). A figura 2 ilustra a arquitetura do protótipo.

Figura 2. Arquitetura do Protótipo.

As subseções seguintes fazem uma descrição da arquitetura do protótipo.

4.1.1 Detectores

Os detectores são responsáveis pela análise do tráfego da rede e geração de alertas, caso alguma anomalia seja encontrada. Na prática, eles não são implementados por este trabalho. A única exigência é que os alertas gerados estejam no padrão IDMEF

(*Intrusion Detection Message Exchange Format*) [Debar *et al.* 2007], uma linguagem de marcação XML para representar dados de alertas e organizá-los de forma hierárquica.

4.1.2 Módulo de Processamento

O Módulo de Processamento é responsável por ordenar cronologicamente os alertas recebidos e traduzi-los (de IDMEF) para o formato suportado (eventos na forma de valores do alfabeto) para o cálculo de episódios frequentes.

Todos os alertas recebidos são traduzidos para um tipo de evento. Os atributos escolhidos para identificar cada alerta são: endereço IP e porta do detector que o gerou, endereço IP e porta do alvo e o tipo de ataque. A tabela 1 representa a correspondência entre eventos e alertas. Neste exemplo, os atributos porta de origem e destino não são usados.

TABELA 1. Exemplo da tabela de correspondência entre eventos e alertas

Evento	Nome do Evento	IP de Origem	IP de Destino
A	Community SIP DNS no such name	172.16.112.100	172.16.115.20
B	NETBIOS NT NULL session	172.16.112.100	172.16.112.20
C	ATTACK-RESPONSE directory listing	172.16.112.194	172.16.112.100
D	NETBIOS NT NULL session	172.16.112.20	172.16.112.100
E	ATTACK-RESPONSE Invalid URL	172.16.113.148	207.200.75.201
…	…	…	…
T	ATTACK-RESPONSE 403 Forbidden	172.16.113.204	137.245.85.134

A marca de tempo (*timestamp*), outro atributo extraído de cada alerta, é usado em associação com a tabela 1 para construir uma lista dos eventos ordenada temporalmente (em segundos), como mostra a figura 3.

```
 1  A I
 2  C T F C
 3  B O
 7  A X K M
15  A L P J
19  A S H U L R J I
25  B
27  A
```

Figura 3. Sequência de eventos realizada pelo módulo de ordenação.

4.1.3 Módulo FED

O Módulo FED é responsável pelo cálculo dos episódios frequentes e geração das regras baseadas nesses episódios. Em relação ao funcionamento, o módulo recebe uma lista contendo os tipos de eventos e calcula os episódios frequentes de acordo com o

tamanho da janela de tempo estipulado e do limite de frequência estabelecido. Para tanto, o módulo FED é dividido em quatro funções:

- **Coletor de Eventos**: verifica a lista que contém os tipos de eventos (figura 3) e identifica aqueles que são mais frequentes. Basicamente, recebe uma lista de episódios do mesmo tamanho como entrada e descobre quais entre esses são frequentes, verificando se cada episódio está contido na sequência de eventos globais. Como resultado, todos os episódios frequentes são devolvidos.
- **Gerador de Candidatos**: recebe os episódios frequentes de tamanho X e gera uma nova lista de possíveis candidatos a episódios frequentes de tamanho X+1. Basicamente, para cada entrada recebida, todos os candidatos a episódios frequentes são calculados. Para melhorar o entendimento dessa função, um exemplo é apresentado. Dada uma lista de episódios frequentes formada por (AA, AB, AC, AD), os possíveis candidatos do episódio AB seria (ABA, ABB, ABC, ABD, ..., ABZ) e cada um deles será testado para gerar possíveis candidatos. Para o primeiro candidato, ABA, os subconjuntos originados são AB, BA e AA. Então, para provar que o candidato ABA é frequente, seus subconjuntos (AB, BA e AA) são comparados com a lista de episódios frequentes. Como resultado, a ABA é um possível candidato, pois AB, BA e AA são representados na lista por episódios frequentes pelos episódios AA e AB. Por outro lado, o candidato ABZ não é uma vez que os subconjuntos AZ e BZ não têm representação na lista de episódios frequentes.
- **Gerador de Episódios Frequentes**: calcula todos os episódios frequentes, utilizando as funções do gerador de candidatos e do coletor de eventos, retornando os episódios frequentes.
- **Gerador de Regras**: gera regras baseando-se nos episódios frequentes enviados pelo gerador de episódios frequentes. O algoritmo 1 ilustra a função de geração de regras.

Algoritmo 1. Algoritmo simplificado do gerador de regras

Entrada: sequência de eventos **E**, tamanho da janela **win**, frequência **fr** e confiança **conf**

```
01: regras = []
02: /* Encontrar os episódios frequentes */
03: EpisódiosFrequentes = calcularEpisódiosFrequentes (E, win, fr)
04: /* Geração de regras */
05: for all episódio in EpisódiosFrequentes do
06:     for all subepisódio in episódio do
07:         if frequência(episódios)/frequência(subepisódios) ≥ conf then
08:             regras += [episódios, subepisódios, confiança(episódios / subepisódios)]
09:         end if
10:     end for
11: end for
12: return regras
```

O algoritmo inicia calculando os episódios frequentes (linha 3) para uma dada sequência de eventos *E* e usando uma janela de tempo de tamanho *win* e um limite de frequência *fr*. Como resultado, uma lista contendo todos os episódios frequentes é retornada (*EpisódiosFrequentes*). Em seguida, o algoritmo executa o processo de geração das regras através de uma série de iterações. A primeira delas extrai os

episódios que compõem a lista de episódios frequentes (linha 5). A segunda extrai os subepisódios relativos aos episódios pais (linha 6). Depois, de posse do episódio e seus subepisódios, o algoritmo testa se a relação (proporção) entre um episódio e seus subepisódios é maior ou igual ao limite de confiança definido *conf* (linha 7). Se o resultado for verdadeiro, então uma nova regra é gerada (linha 8). O algoritmo termina retornando todas as regras geradas. Atualmente, os valores de *win*, *fr* e *conf* são atribuídos manualmente.

Redução de Regras

Tipicamente, a descoberta de episódios frequentes gera um número elevado de episódios e consequentemente de regras, dentre as quais muitas são redundantes ou repetidas. Para resolver essa ineficiência, duas técnicas de redução, propostas por [Qin and Hwang 2004], para reduzir o espaço de regras e fornecer uma visão simplificada dos padrões de dados são utilizadas. A ideia é estabelecer se uma regra é eficaz (frequentemente utilizada) ou ineficaz (raramente utilizada). O algoritmo 2 ilustra essas técnicas.

Algoritmo 2. Algoritmo simplificado para redução de regras

Entrada: regras ***r***

```
01: regrasReduzidas = []; regrasNovas = [];
02: for all regras in r do
03: /* Aplicação da Transposição */
04:     regrasNovas += Transposição(regras);
05: end for
06: for all regras in regrasNovas do
07: /* Aplicação da Eliminação de Redundantes*/
08:     regrasReduzidas += EliminaçãodeRedundantes(regras);
09: end for
10: return regrasReduzidas
```

A primeira regra, *transposição*, afirma que dadas duas regras ($A \rightarrow AAAA\ e\ A \rightarrow AAAA$), que descrevem o comportamento do evento A, a primeira é vista como mais efetiva do que a segunda porque satisfaz a lei da transposição (a segunda regra pode ser deduzida da primeira). Assim, a primeira regra é mantida e a segunda removida. De forma geral, o objetivo é ter o menor lado esquerdo possível de cada regra, devido ao simples fato de que regras curtas são mais fáceis de aplicar ou comparar. A segunda regra, *eliminação*, também assume que regras com o menor lado esquerdo são mais efetivas. Deste modo, se existir duas regras ($A \rightarrow B\ e\ B \rightarrow C$) no conjunto de regras e existir uma regra muito frequente ($A \rightarrow BC$), pode-se assumir que a regra ($A \rightarrow BC$) é redundante, já que ela pode ser construída pelas duas regras anteriores. Assim, as duas primeiras regras são mantidas e a última removida.

5. AVALIAÇÃO E RESULTADOS

Esta seção descreve o processo de avaliação de desempenho da solução propostas. Para tanto, o processo de avaliação foi dividido em dois experimentos. No primeiro, o *DARPA 2000 Intrusion Detection Scenario Specific Data Set* [MIT 2000], uma base de tráfego bastante conhecida na literatura e empregada em dezenas de trabalhos de avaliação de alertas, foi utilizado. A ideia é avaliar a precisão da correlação dos alertas

das anomalias existentes no cenário, detectando, desta forma, verdadeiros e falsos alertas. Além disso, essa base contém uma grande quantidade de tráfego reconhecido como anômalo o que fornece uma análise qualitativa da solução.

No segundo experimento, tráfego real capturado na rede do Grupo de Pesquisa em Redes de Computadores (GPRT) do Centro de Informática (Cin) da Universidade Federal de Pernambuco (UFPE) foi utilizado com o objetivo de validar a eficiência e precisão da solução em ambiente real. Em ambos os experimentos foi utilizado o IDS Snort versão 2.8.6 (Snort 2009) para geração dos alertas.

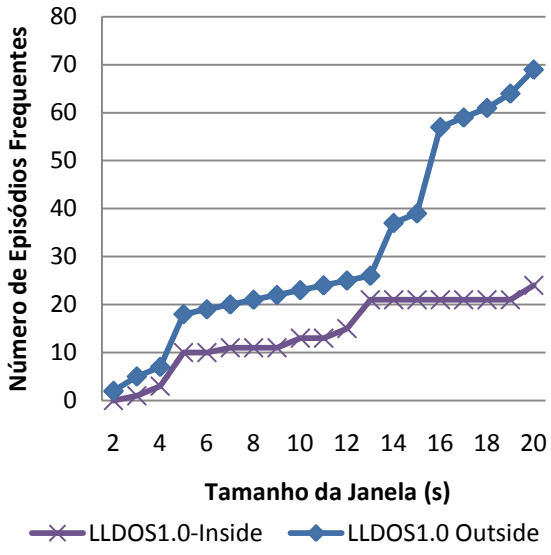
5.1 DARPA 2000

O DARPA 2000 dataset é uma conhecida base de informação para avaliação de IDSs, contendo dois cenários (LLDOS 1.0 e LLDOS 2.0.2), onde o tráfego coletado pertence tanto a uma rede externa (*outside*), no caso uma zona desmilitarizada (do inglês *DeMilitarized Zone* – DMZ), quanto a uma rede interna (*inside*).

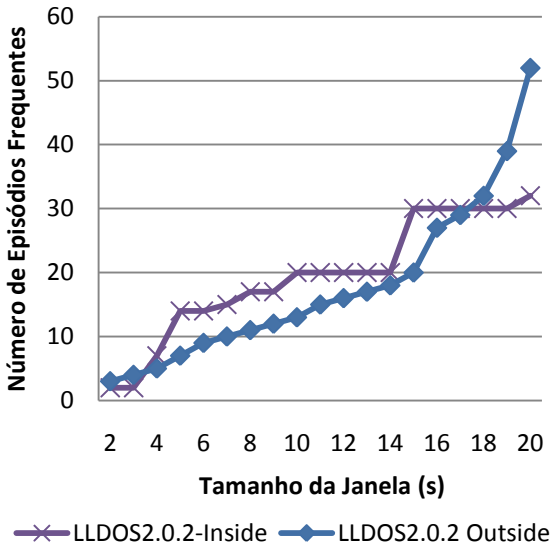
5.1.1 Desempenho

Para testar o desempenho do protótipo, os cenários LLDOS 1.0 e 2.0.2 (tanto *inside* quanto *outside*) foram utilizados para avaliar a influência do tamanho da janela e do limite de frequência na geração de episódios frequentes.

O cenário LLDOS 1.0, composto por uma sequência de 1109 e 2.465 alertas (*inside* e *outside*, respectivamente) e com quase três horas de duração, apresenta 602 e 90 tipos diferentes de eventos, respectivamente, com frequência e distribuições diversificadas. Em média, um alerta é recebido a cada minuto, mas ocorrem rajadas de até 169 alertas em um único segundo. Dentro do cenário LLDOS 2.0.2, existem 935 (*inside*) e 1108 (*outside*) alertas em um período de quase 1 hora e 30 minutos, onde existem 867 e 41 tipos diferentes de eventos, respectivamente. A figura 4 (a e b) apresenta os efeitos do tamanho da janela de tempo no número de episódios frequentes para ambos os cenários.



(a) Cenário LLDOS 1.0



(b) Cenário LLDOS 2.0.2

Figura 4. Relação entre o número de episódios frequentes e o tamanho da janela de tempo (com limite de frequência 0.002)

Tendo em vista que a descoberta de episódios frequentes está relacionada com a existência de possíveis candidatos, é correto afirmar que quanto maior o tamanho da janela, maior será o número de candidatos e, consequentemente, o número de episódios frequentes descobertos. O tempo de processamento e a participação também aumentam da mesma forma.

Antes de apresentar os resultados relativos a regras de episódios, é necessário esclarecer a importância e utilização do limite de frequência em qualquer análise. O conceito de frequentes, representado pela imagem do limiar de frequência, é essencial para a descoberta de todos os episódios frequentes de uma sequência e, consequentemente, a obtenção de regras que descrevem as conexões entre os eventos. Dessa forma, valores típicos dos limites de frequentes para análise são: 0,001, 0,002, 0,005, 0,01, 0,02, 0,05 e 0,1 [Manilla *et al.* 1997].

De fato, a maioria dos trabalhos utilizando FED emprega os valores do limite de frequentes igual ou superior a 0,01. A explicação é simples. Pequenos valores do limiar de frequência, tais como 0,001 e 0,002, permitem a geração de uma enorme quantidade de candidatos e episódios frequentes, com impactos relevantes sobre o tempo de processamento. Por exemplo, no cenário LLDOS2.0.2 *inside*, a utilização do valor do limite de frequentes 0,001, com o tamanho da janela de tempo igual a 12, gera 753.424 possíveis candidatos de tamanho 2, durante um período de 25 minutos de processamento.

5.1.2 Regras de Episódios

A fim de testar a geração de regras de episódio, dois parâmetros são utilizados: o tamanho da janela de tempo e o limite de confiança. O primeiro tem influência sobre o número de episódios frequentes gerados e, consequentemente, o número de regras. O segundo tem impacto na qualidade das regras. Quanto maior o limite de confiança, melhor é a qualidade e a confiabilidade das regras obtidas.

A Figura 5 ilustra o crescimento do conjunto de regras geradas em relação ao tamanho da janela tempo nos dois cenários DARPA 2000. Foram assumidos um limite de confiança de 0,6 e um limite de frequência de 0,005. Estes valores foram atribuídos manualmente através de testes que apontaram uma quantidade adequada de regras. Valores menores do limite de frequência geram volumes grandes de episódios enquanto valores menores do limite de confiança permitem a geração de um grande número de regras. Consequentemente, o uso desses dois parâmetros com valores pequenos acabam por mascarar a presença de falsos positivos e falsos negativos na análise.

A Figura 5 ilustra a relação entre a quantidade de regras geradas quando se varia o tamanho da janela de tempo. Em cada cenário é realizada uma comparação entre as abordagens com e sem a técnica de redução de regras. O cenário LLDOS 1.0 *inside* teve uma redução de 8% (Figura 5a), enquanto o LLDOS 2.0.2 *inside* teve uma redução variando entre 24 e 29% (Figura 5c). Tal fato é explicado pelas características dos ataques, que geraram muitos eventos. Por conseguinte, esta grande diversidade (602 eventos de 1.109 alertas no LLDOS 1.0 e 867 eventos de 935 alertas no LLDOS 2.0.2) é refletida pela baixa descoberta de episódios frequentes e consequente geração de regras. Por outro lado, os cenários *outside* apresentam reduções consistentes e relevantes. No

LLDOS 1.0, a variação entre as regras varia entre 33% e 73% (Figura 5b), enquanto que no LLDOS 2.0.2 esta variação ficou entre 33% e 81% de regras (Figura 5d).

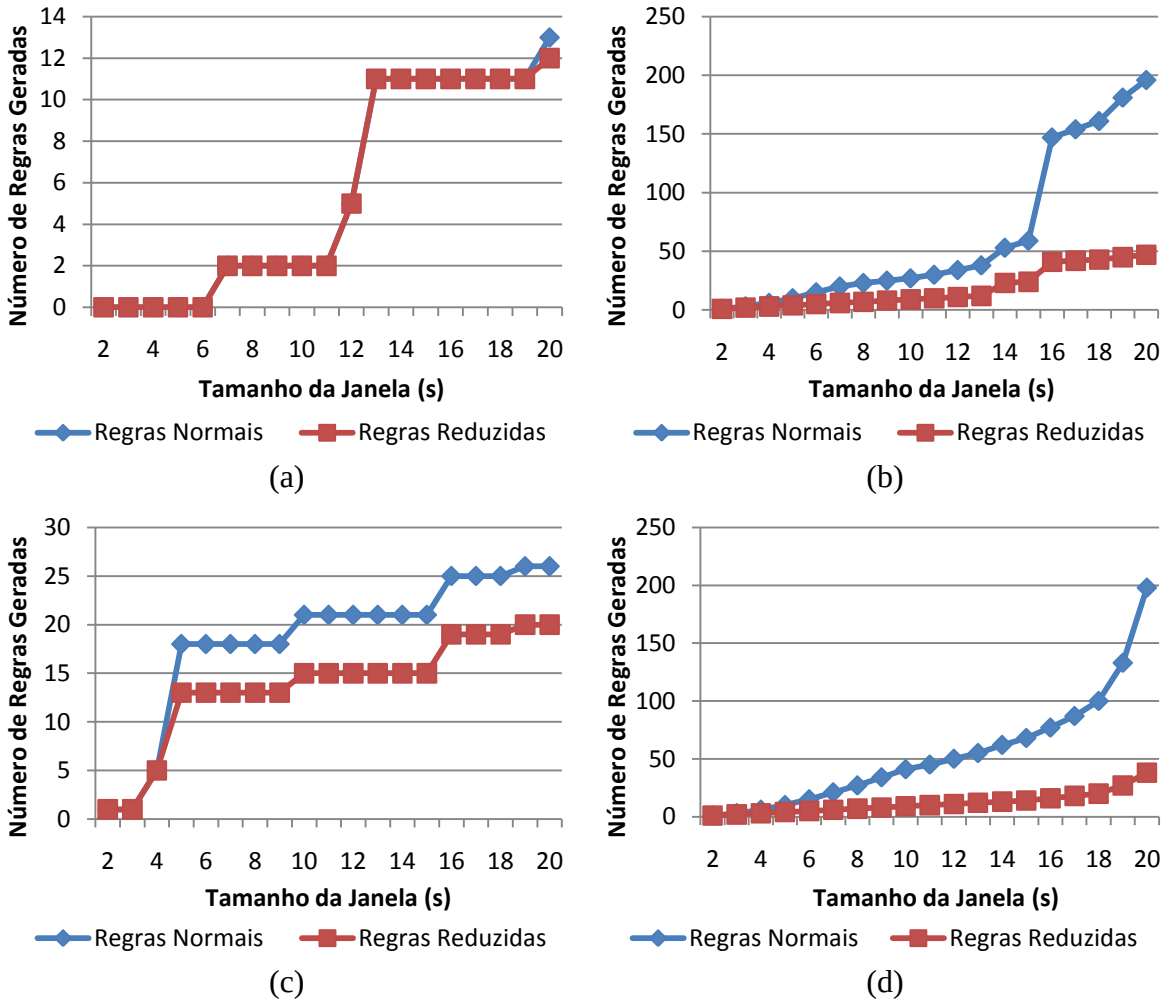


Figura 5. Regras geradas para os cenários LLDOS 1.0 e LLDOS 2.0.2

5.2 GPRT

As instalações do Grupo de Pesquisa em Redes e Telecomunicações (GPRT) da Universidade Federal de Pernambuco (UFPE) foram utilizadas para simular o comportamento de um ataque com múltiplos passos (do inglês *multi-step attack*).

O aspecto interessante e relevante de ataques com múltiplos passos é que eles podem e devem ser observados por diferentes detectores, o que é ideal para provar a eficácia da solução proposta. Contudo, é necessário "juntar todos os pedaços" para que um cenário de ataque possa ser visto como um ataque com múltiplos passos.

Para realizar este experimento, o *worm* Blaster (CERT 2003) foi escolhido e seu comportamento foi simulado. Basicamente, o cenário de ataque consiste das seguintes fases. Na primeira, o atacante realiza uma varredura na rede, procurando por computadores com a porta TCP 135 aberta. Na segunda, o atacante tentar explorar a vulnerabilidade *DCOM RPC vulnerability in Microsoft Windows*, fazendo com que a vítima habilite um *backdoor* na porta TCP 4444. Na terceira fase, o atacante instrui as vítimas a copiarem, via TFTP (UDP porta 69) o arquivo MBLASTER.EXE, localizado

no computador atacante, e a também o executarem, infectando assim seu sistema. Uma quarta fase, simulando uma conexão da máquina infectada com sítios Web responsáveis pela divulgação do *worm* Storm (Symantec 2007), também foi inserida para melhorar a visualização das etapas do ataque.

Um cenário de teste composto por 9 computadores (sendo 1 atacante e 8 vítimas) com sistema operacional Linux e utilizando uma série de scripts para simular todas as fases foi montado. O experimento foi executado no dia 25 de Junho de 2010, repetidamente, das 10 às 18 horas. Para coletar o tráfego desse experimento foi utilizado o Snort IDS (versão 2.8.3.2).

5.2.1 Análise do tráfego

O ataque com o *worm* Blaster gerou 40.200 alertas, correspondendo basicamente a apenas 3 tipos de eventos: *TCP Portsweep* (fase 1), *Blaster Backdoor* (fase 2) e *TFTP Get* (fase 3). A tabela 2 exemplifica alguns alertas e tipos de eventos deste experimento.

Tabela 2. Exemplos de tipos de eventos para o ataque do *worm* Blaster

Evento	Nome do Evento	IP de Origem	IP de Destino
A	TCP Portsweep	192.168.0.96:1412	192.168.0.50:135
B	TCP Portsweep	192.168.0.96:4055	192.168.0.51:135
H	TCP Portsweep	192.168.0.96:10001	192.168.0.57:135
I	Blaster Backdoor	192.168.0.96:34521	192.168.0.50:4444
J	Blaster Backdoor	192.168.0.96:50674	192.168.0.51:4444
H	Blaster Backdoor	192.168.0.96:12543	192.168.0.57:4444
Q	TFTP Get	192.168.0.50: 5643	192.168.0.96:69
R	TFTP Get	192.168.0.51: 3027	192.168.0.96:69
S	TFTP Get	192.168.0.57: 3027	192.168.0.96:69
E1	Storm worm phone address	192.168.0.50: 64267	222.252.232.184:22861
F1	Storm worm phone address	192.168.0.51: 4530	216.139.142.17:10788
L1	Storm worm phone address	192.168.0.57: 1155	217.77.54.253:12358

Usando os parâmetros limite de frequência igual a 0.01, o tamanho da janela igual a 10 e o limite de confiança igual 0.8, a tabela 3 apresenta o cálculo dos episódios frequentes.

Tabela 3. Cálculo dos Episódios Frequentes para o cenário do *worm* Blaster

Tamanho da Janela	Candidatos	Episódios Frequentes	Participation (%)
1	32	32	100.00%

2	289	153	52.94%
3	1376	612	44.47%
4	3468	1428	41.17%
5	5712	2142	37.50%
6	6426	2142	33.33%
7	4998	1428	28.57%
8	2652	612	23.07%
9	918	153	16.66%
10	187	32	17.11%

É possível notar que a presença de um baixo número de alertas gera um número menor e mais focado de episódios frequentes. Tal afirmação é provada pelo resultado final de episódios frequentes descobertos (32) para o tamanho máximo da janela de tempo.

A próxima etapa da FED é a geração de regras de episódios. De posse dos episódios frequentes, foram geradas 4.334 regras que atendiam aos parâmetros de limite de frequência (0.01) e limite de confiança (0.8). Aplicando as técnicas de redução de regras (algoritmo 2), o número de regras foi reduzido a 137, o que tornou possível a identificação do ataque com múltiplos passos. A regra número 7 (*A* $\rightarrow$ *AI com confiança 1.00*) indica que sempre após um evento *A* (*TCP Portsweep*, com origem 192.168.0.96:1412 e com destino 192.168.0.50:135) ocorre o evento *I* (*Blaster Backdoor*, com origem 192.168.0.96:34521 e com destino 192.168.0.50:4444) com precisão de 100%. Já a regra 36 mostra que sempre após um evento *I* ocorrem os eventos *Q* (*TFTP Get*, com origem 192.168.0.50:5643 e com destino 192.168.0.96:69) e *E1* (*Storm worm*, com origem 192.168.0.50:64267 com destino 222.252.232.184:22861) também com confiança de 100%.

Desta forma, é fácil deduzir que, com uma confiança que 100% dos casos, quando um evento *A* acontece, a sequência de eventos *AIQE1* (*TCP Portsweep*, *Blaster Backdoor*, *TFTP Get* e *Storm worm*) também ocorre, provando a existência de um ataque com múltiplos passos.

É importante ressaltar que a solução proposta neste artigo não é capaz de identificar a sequência de eventos apresentada acima como um ataque com múltiplos passos. Tal tarefa ainda cabe a um especialista de segurança. Contudo, percebe-se que a descoberta de sequências de eventos suspeitas ou anômalas é um passo importante no processo de detecção e mitigação de atividades maliciosas no tráfego da rede. Por exemplo, na medida que tais sequências são identificadas e comprovadas como pertencentes a ataques, elas podem ser armazenadas em uma base de dados para serem empregadas em futuras comparações e possivelmente na detecção de ataques.

6. Conclusão

Atualmente, anomalias continuam causando inúmeros prejuízos a empresas e instituições. Ataques de negação de serviço, *scans*, *worms*, vírus e outros tipos de males ainda geram problemas a milhares de administradores de rede e até mesmo a usuários comuns. Apesar do desenvolvimento e aperfeiçoamento das técnicas de detecção e predição de anomalias, muito do tráfego que circula na rede ainda é malicioso, causando prejuízos econômicos e financeiros. Desta forma, esse trabalho contribuiu com a implementação de um protótipo capaz de, usando a técnica de episódios frequentes, correlacionar alertas e gerar regras que representam ataques e anomalias de tráfego, permitindo uma rápida e mais correta detecção e possível mitigação de tais eventos na rede.

Este artigo apresentou, em detalhes, a técnica de descoberta de episódios frequentes utilizando uma abordagem teórica. Em seguida, relacionou e descreveu os principais trabalhos relacionados desta técnica com detecção de anomalias. Também foram apresentadas a arquitetura e a implementação de um protótipo capaz de detectar (confirmar) a presença de anomalias.

Por fim, testes para avaliação de desempenho e eficácia foram realizados utilizando o famoso conjunto de dados DARPA 2000 e um ataque simulado em ambiente controlado (GPRT). Os resultados indicam a eficácia do uso de episódios frequentes. Ficou provado que através das regras geradas pelo protótipo também foi possível detectar padrões que podem ser utilizados para predição de anomalias.

Resumindo, a aplicação de episódios frequentes na detecção de ataques mostrou-se útil. O papel dos diferentes parâmetros da técnica, tais como tamanho da janela e limite de frequência foi demonstrado nos cenários de teste. Por fim, fica claro que uma possível interação entre este protótipo e soluções de segurança (como Firewall) certamente aumentará a eficácia das soluções de segurança e fornecerá a base para uma solução mais automatizada de gerenciamento de segurança.

No diz que respeito a trabalhos futuros, pretende-se:

1. Testar novos experimentos para melhor validar o protótipo;
2. Armazenar as regras para a criação de uma base de anomalias, permitindo uma fácil e rápida comparação de eventos atuais com os eventos antigos;
3. Implementar um módulo para tomada de ações baseados nas regras geradas para episódios previstos de ocorrer (com elevada confiança);
4. Implementar outros tipos de técnicas de correlação e predição de eventos, de forma a maximizar a taxa de acertos nas anomalias das redes.
5. Implementar um mecanismo de configuração automática dos parâmetros janela de tempo, limite de frequência e limite de confiança.

Referência

CERT. (2003) “CERT Advisory CA-2003-20 W32/Blaster worm,” http://www.cert.org/advisories/CA-2003-20.html.

Debar, H., Curry, D., and Feinstein, B. (2007) "The Intrusion Detection Message Exchange Format (IDMEF)", RFC 4765, Março.

Feitosa, E. L., Souto, E., e Sadok, D. (2008) “Tráfego Internet não Desejado: Conceitos, Caracterização e Soluções”, In Livro-texto de mini-cursos do VIII Simpósio Brazileiro de em Segurança da Informação e Sistemas de Computação (SBSeg’08). Porto Alegre, Brasil: SBC, ch. 3.

Hwang, K., Cai, M., Chen, Y., and Qin, M. (2007) "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes”, *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 1, páginas 41-55.

Lee, W., Stolfo, S. J., and Mok, K. (2000) "Adaptive Intrusion Detection: A Data Mining Approach", *Artificial Intelligence Reviews*, Vol. 14, No. 6, páginas 533-567, December.

Mannila, H., Toivonen, H., e Verk, A. I. (1997) “Discovery of Frequent Episodes in Event Sequences", *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, páginas 259-289.

MIT. (2000) “2000 DARPA Intrusion Detection Scenario Specific Data Sets”, http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html.

Qin, M. e Hwang, K. (2004) "Frequent Episode Rules for Intrusive Anomaly Detection with Internet Data mining", In *USENIX Security Symposium*, páginas 1-15.

Snort. (2009) “Snort”, http://www.snort.org.

Soleimani, M. e Ghorbani, A. A. (2008) "Critical Episode Mining in Intrusion Detection Alerts", In *Proceedings of the Communication Networks and Services Research Conference (CNSR)*, páginas 157-164.

Symantec. (2007) “Outbreak alert: storm trojan”, http://www.symantec.com/outbreak/storm_trojan.html. 2007.

Ye, N., Chen, Q., e Borror, C. M. (2004) "EWMA Forecast of Normal System Activity for Computer Intrusion Detection", *IEEE Transactions on Reliability*, Vol. 53, páginas. 557- 566.