Capítulo 4

Análise e Visualização de Logs de Segurança

André Ricardo Abed Grégio[1], Rafael Santos[2].

Resumo

Sistemas computacionais geram grande quantidade de registros, ou logs*, de suas atividades, os quais podem ser utilizados para prover informações sobre o funcionamento do sistema, monitorar tráfego da rede, tentar identificar a atuação de software malicioso (*malware*), entre outras aplicações e funções. A análise destes* logs *costuma ser impraticável para analistas humanos devido ao seu volume. Uma forma de encontrar informações importantes em grandes massas de dados com várias dimensões é a aplicação de técnicas de visualização. Neste capítulo, veremos os conceitos de técnicas de visualização com aplicações em visualização de dados (eventos) relacionados a área de segurança da informação, com exemplos práticos comentados.*

Abstract

Computer systems generates vast amounts of logs of their activities, which can be used to provide information about its working, to monitor network traffic, to help the identification of malware activities and for other applications and functions. Analysis of these logs can be infeasible for human analysts due to its sheer volume. One way to find important information in large amounts of (often multidimensional) data is to apply visualization techniques to it. In this chapter we will see the basic concepts of visualization techniques with applications on information systems security data visualization, showing and commenting practical applications.

4.1. Introdução

Sistemas computacionais interligados em redes possuem como uma de suas características básicas a geração de registros das atividades decorrentes da interação dos seus dispositivos componentes – sistema operacional, aplicações, equipamentos de rede (switches,

[1]Divisão de Segurança de Sistemas de Informação, Centro de Tecnologia da Informação Renato Archer.

[2]Laboratório Associado de Computação e Matemática Aplicada, Instituto Nacional de Pesquisas Espaciais.

roteadores), mecanismos de defesa (*firewalls*, *Intrusion Detection Systems*, antivírus), isto é, hardware e software em geral. Tais registros, ou *logs*, têm o intuito de prover informações sobre o funcionamento das partes do sistema, permitir a monitoração do tráfego da rede em busca de eventos suspeitos ou falhas, tentar identificar a atuação de software malicioso (*malware*) em computadores ou redes, identificar a proveniência e/ou conteúdo de mensagens de e-mail não solicitadas, bem como auxiliar na auditoria em caso de incidentes envolvendo a segurança de sistemas de informação.

Devido ao grande número de sistemas gerando *logs*, há uma imensa quantidade de dados que necessitam ser armazenados e, principalmente, analisados de modo a prover alguma informação que possa ajudar na identificação de um determinado tipo de evento de segurança. A análise destes dados costuma ser impraticável para analistas humanos, bem como pode não gerar resultados apresentáveis do ponto de vista de inteligência dependendo das ferramentas que forem utilizadas para extração de informações.

Uma forma efetiva de encontrar informações importantes em grandes massas de dados com várias dimensões é vendo figuras que correspondem a estes números [20], ou seja, aplicando de técnicas de visualização. A área de visualização de dados aplicada à segurança ainda está bastante recente tendo, portanto, uma ampla gama de ferramentas e aplicações a se explorar – adicionalmente muitos problemas pedem soluções *ad hoc*, que devem ser integradas a sistemas específicos, justificando portanto o estudo de técnicas de visualização e sua aplicação à segurança, pois não existem soluções prontas que atendam à todas as necessidades.

O objetivo deste curso é apresentar os conceitos de técnicas de visualização com aplicações em visualização de dados (eventos) relacionados a área de segurança da informação com foco em análise de tráfego de rede suspeito ou malicioso, identificação de padrões de ataque baseados em dados temporais através de *logs* de sensores de monitoração da Internet no ciberespaço nacional, classificação de *malware* utilizando logs de coletores de *malware* e analisando como eles são disseminados e quais serviços são atacados, entre outros assuntos relacionados.

O foco dos exemplos deste curso será a visualização de *logs* relacionados a dados de ataques a honeypots, fluxos de tráfego de rede, coletores de *malware* e de *spam*. Nestes *logs*, as informações relevantes vão desde a data e hora dos eventos, passando pela origem do ataque, dispositivo alvo, tráfego de rede, conteúdo do tráfego e até a informação geográfica do atacante ou localização do repositório do código malicioso disseminado.

4.2. Conceitos de Visualização

Edward Tufte, professor emérito da Universidade de Yale, escreve em seu livro clássico *The Visual Display of Quantitative Information* [20] que *"... gráficos sobre dados podem fazer muito mais do que simplesmente ser substitutos para pequenas tabelas estatísticas. Na sua melhor concepção, gráficos são instrumentos para compreender informação quantitativa. Frequentemente a forma mais efetiva de descrever, explorar e sumarizar um conjunto de números – mesmo um conjunto com muitos números – é ver figuras destes números. Adicionalmente, de todas as formas de analisar e comunicar informação estatística, gráficos bem feitos sobre dados são geralmente ao mesmo tempo a mais simples e mais poderosa"*. Este comentário define bem o papel de visualização na análise de

dados: ver “figuras dos números” pode explicar muito mais sobre eles do que medidas estatísticas.

Visualização é uma tarefa realizada com muita facilidade por humanos: o sistema visual humano é capaz de identificar padrões, exceções, tendências, relações entre objetos percebidos visualmente mesmo que não seja possível descrever estes fenômenos em linguagem natural. Esta capacidade aliada ao uso de computadores para preparar, organizar e visualizar dados possibilita a exploração de dados de formas novas, eficientes e interessantes.

Visualização de dados (em particular de dados técnico-científicos, como dados relacionados com sistemas de segurança) pode ser usada para vários objetivos relacionados à análise destes dados: [12]

- **Análise Exploratória**, usando dados de natureza conhecida sem uma hipótese definida sobre fenômenos que podem ocorrer nestes dados. Geralmente envolve a busca visual por tendências, exceções, estruturas, etc. nos dados. O resultado da análise exploratória pode ser a definição de hipóteses.

- **Análise para Confirmação**, usando dados de natureza conhecida e hipóteses sobre fenômenos relacionados a estes dados. Através de visualização a hipótese pode ser confirmada ou rejeitada.

- **Apresentação**, usando os dados, com um objetivo para a demonstração dos dados, fenômenos relacionados a estes ou hipóteses. Deve-se definir uma técnica para apresentação apropriada e que permita fácil interpretação.

4.2.1. Tipos de técnicas de visualização

Existem muitas técnicas de visualização de dados, desde as mais simples e genéricas como gráficos simples de área, torta ou pizza, linhas, histogramas, pontos, etc. (que estão implementadas praticamente em todas planilhas eletrônicas) até as mais complexas e específicas como fatiamento (*slicing*) de volumes em três dimensões para apresentação de imagens bi-dimensionais, como as usadas para visualização de imagens tri-dimensionais como as de ressonância magnética nuclear. Nesta seção apresentaremos algumas técnicas intermediárias que tem características que as tornam interessantes para a visualização de dados relacionados a segurança.

Muitas técnicas de visualização podem ser agrupadas em algumas categorias, sendo que algumas técnicas podem pertencer a mais de uma categoria ou mesmo a nenhuma delas. Por causa da vasta quantidade de técnicas de visualização somente algumas categorias e exemplos serão apresentados. Algumas das categorias apresentadas nesta seção são baseadas em tutoriais e notas de aulas de Daniel Keim [12].

Técnicas geométricas são técnicas que permitem a visualização dos dados através de transformações geométricas (ex. reorganizações, projeções) dos valores dos seus atributos. Uma das técnicas geométricas mais conhecidas é a matriz de espalhamento (*scatterplot matrix*) [4], que apresenta uma matriz bidimensional de plotagens bidimensionais de dados, onde cada combinação de duas dimensões é representada por uma plotagem. Este

tipo de visualização é mostrado na Figura 4.1[3], onde alguns atributos de dados de tráfego de rede suspeito e inofensivo são comparados [8].

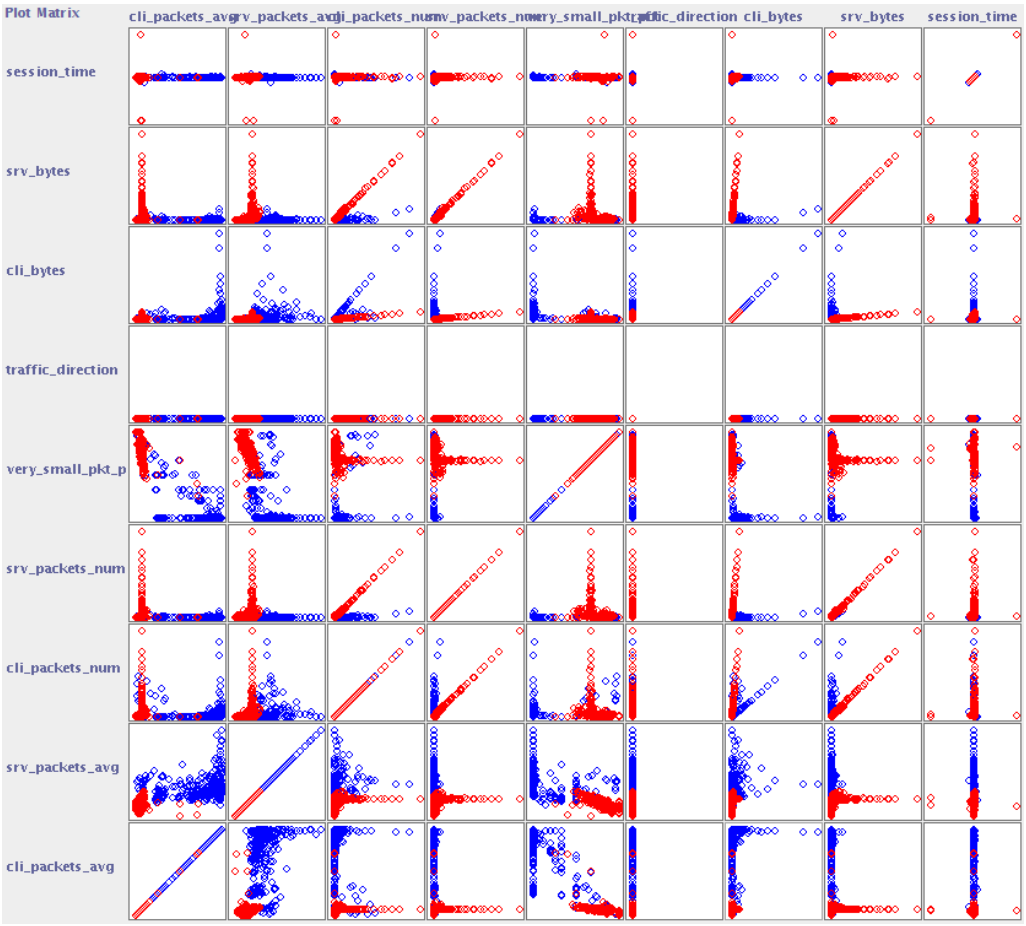

Figura 4.1. Visualização de tráfego de rede suspeito e inofensivo com matrizes de espalhamento.

Em visualizações com matrizes de espalhamento como a mostrada na Figura 4.1 podemos observar correlações diretas e inversas entre atributos (dois a dois), distribuição de valores nos atributos, padrões, exceções e outros fenômenos que podem servir para sugerir hipóteses sobre os dados.

Outra técnica geométrica que permite a visualização de correlações entre atributos, padrões e exceções é a de coordenadas paralelas [9, 10]. Nesta técnica criamos um eixo para cada um dos atributos, organizamos estes eixos de forma paralela e equidistante, e para cada dado a ser visualizado traçamos uma linha poligonal que cruza os eixos na altura dos valores correspondentes para aqueles atributos. Um exemplo deste tipo de visualização é mostrado na Figura 4.2, que mostra os mesmos dados usados para criar a Figura 4.1. Neste tipo de gráfico é possível inferir métricas para separação de tráfego possivelmente ferindo as políticas de segurança da instituição, pois as diferenças do tipo de tráfego (suspeito ou inofensivo) podem ser claramente observadas nos eixos de alguns

[3]As figuras coloridas correspondentes às mostradas neste capítulo podem ser encontradas em `http://www.lac.inpe.br/~rafael.santos/cotb2010.jsp`.

dos atributos. Na figura, podemos ver que o primeiro e o quinto eixo poderiam ser cortados por linhas horizontais e, desta forma, teríamos uma boa separação do tráfego suspeito e inofensivo.

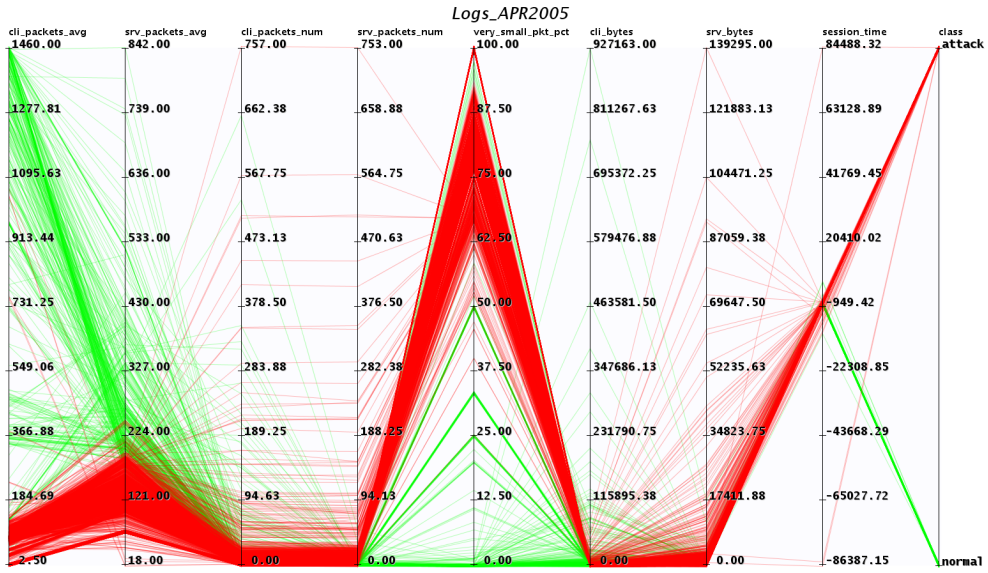

Figura 4.2. Visualização de tráfego de rede suspeito e inofensivo com coordenadas paralelas.

Ainda outra técnica geométrica é baseada na reorganização dos dados multidimensionais em uma grade regular em poucas dimensões (tradicionalmente duas) de forma que dados que são próximos no espaço de atributos multidimensional continuam próximos no espaço de dimensões reduzidas. Uma maneira de criar estes gráficos é usando um modelo de rede neural conhecido como Mapa Auto-Organizável de Kohonen (abreviadamente SOM, *Self-Organizing Map*) [14]. Cada elemento da grade regular pode ser usado como componente para visualização dos dados de acordo com a natureza destes dados.

Um exemplo desta técnica de visualização é mostado na Figura 4.3. Nesta figura, pequenas séries temporais (que, com doze medidas, estariam em um espaço de atributos difícil de ser visualizado diretamente) foram agrupadas pelo algoritmo SOM em uma grade de 10×10 células. Cada célula mostra o valor central em cor mais escura e os valores pertinentes à célula em valores mais claros. Com este tipo de técnica de visualização é possível ver quais são os padrões mais frequentes nas séries temporais e quais são as variações em torno destes padrões. As séries temporais foram obtidas dos *logs* de *honeypots* e correspondem ao número de pacotes TCP recebidos por estes *honeypots* a cada cinco minutos. Cada série temporal corresponde então a uma hora de dados coletados pelos *logs*.

Um problema com estas técnicas geométricas pode acontecer se houver uma enorme quantidade de dados a ser visualizada, o que causará o desenho de muitas linhas. Dependendo da ordem em que estas linhas forem desenhadas, algumas podem ser sobrepostas a outras, ocultando estas e impedindo a visualização de alguns dados. Técnicas como transparência e *jitter* (ligeiras perturbações nas coordenadas usadas para desenhar as linhas) podem ser usadas para minimizar este problema.

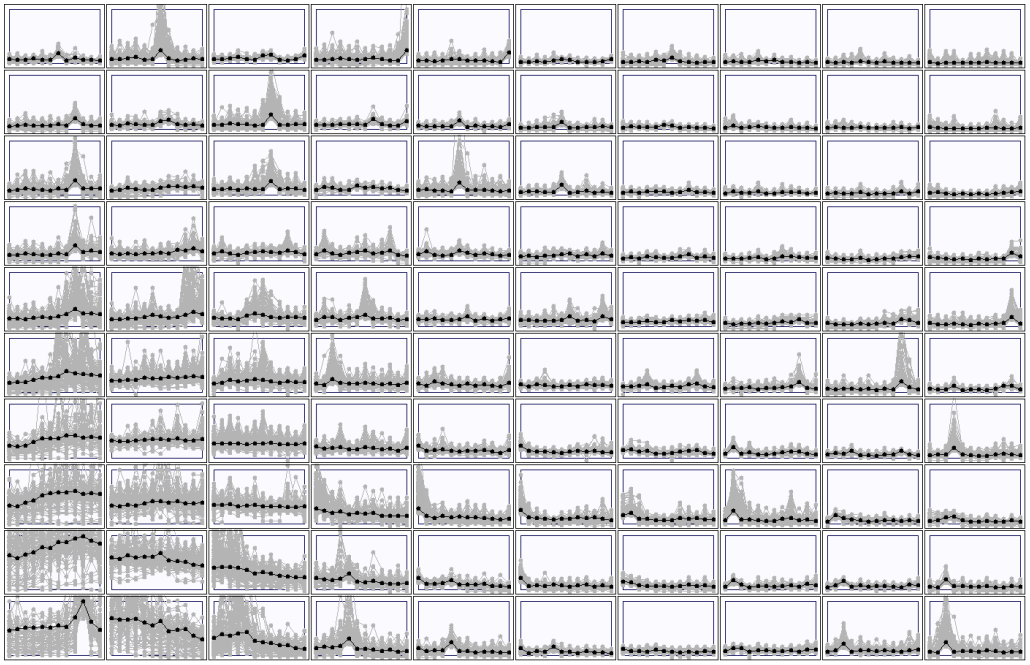

Figura 4.3. Visualização de agrupamentos de séries temporais de acesso à *honeypots*, usando um Mapa Auto-Organizável de Kohonen.

Ainda outras técnicas geométricas de visualização são *Prosection Views* e *Hyperslice* [12]. Estas técnicas são similares às matrizes de espalhamento, mas permitem ao usuário selecionar, em uma das células da matriz, uma região bidimensional dos dados. A composição das regiões bidimensionais posssibilita a seleção de uma região no espaço multidimensional contendo dados de interesse para posterior processamento ou visualização (esta técnica é conhecida como *drill-down*).

Técnicas baseadas em ícones representam os dados multidimensionais a ser plotados como ícones cujas características correspondem a valores dos atributos dos dados. A técnica mais conhecida desta categoria é a *Chernoff faces* [20], que usa pequenos ícones de faces e características como forma da boca, nariz, olhos e posicionamento destes elementos para representar os valores dos dados. Outra técnica baseada em ícones é a codificação por formas (*shape coding*) [1]. Esta técnica mapeia os valores multidimensionais em um pequeno gráfico retangular que é usado como marcador em um gráfico bidimensional, cujas dimensões podem ser espaciais ou temporais. Ícones semelhantes causam uma textura que é aparente no gráfico externo, facilitando a identificação visual de exceções. Uma técnica semelhante a esta é usada na aplicação VisDB [13], que usa cores e arranjo dos ícones em espiral.

Técnicas baseadas em ícones requerem representação adequada para as características dos ícones, que possibilite a rápida interpretação visual através de comparação com outros ícones para identificar similaridades e exceções. Por outro lado, o mapeamento de valores dos atributos para características dos ícones frequentemente requer o uso constante de uma legenda, o que pode confundir o usuário dos gráficos.

Técnicas baseadas em pixels são similares às que usam ícones porque usam pequenos conjuntos em um arranjo espacial para representar os valores multidimensionais,

e organizam estes conjuntos em arranjos maiores que podem representar as dimensões espaciais e/ou temporal do conjunto de dados. Os valores dos atributos são representados por pixels coloridos, geralmente usando um mapa de cores que facilite a identificação de valores similares e diferentes. A interpretação da visualização é novamente feita usando atributos visuais como continuidade e textura para identificar padrões.

Um exemplo bem simples de visualização com uma técnica baseada em pixels é mostrada na Figura 4.4. A figura mostra quantos pacotes TCP, UDP e ICMP foram recebidos por *honeypots* em um período de aproximadamente 10 dias, indicando acessos maliciosos. Pacotes TCP, UDP e ICMP são mostrados em tons de vermelho, verde e azul, respectivamente; com a intensidade da cor correspondendo a um valor normalizado. Os pixels tem bordas cinzas para dias de semana e pretos para fins de semana. O gráfico mostra claramente pequenos intervalos (< 20 minutos) de baixa incidência de tentativas de ataque e picos na incidência destes. Diferentemente de plotagem em séries temporais é possível visualizar um intervalo de tempo bem maior, às custas da indexação temporal explícita.

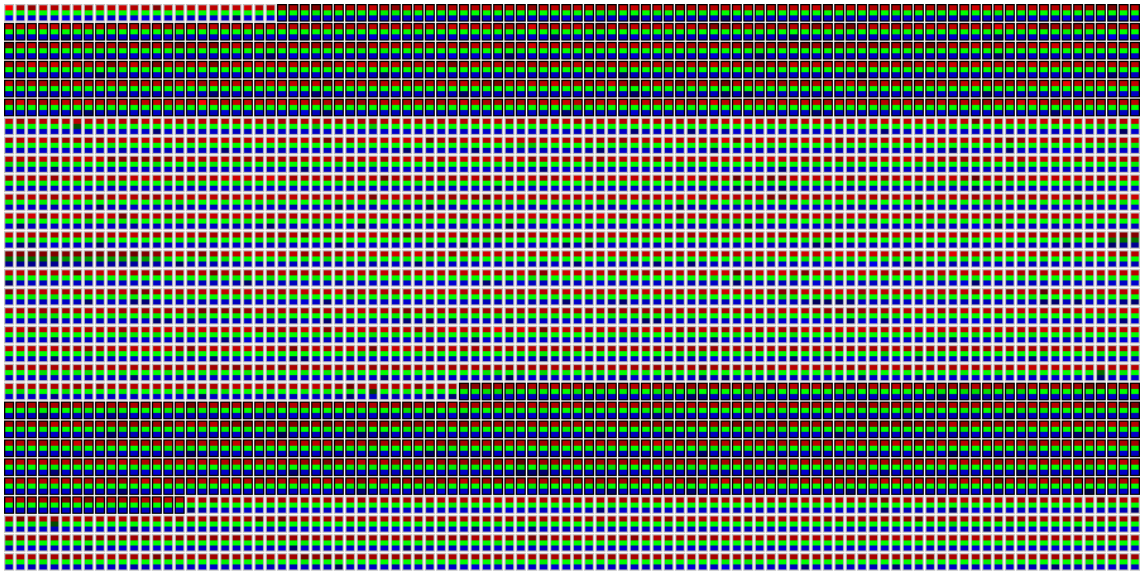

Figura 4.4. Visualização de tipos de pacotes enviados a uma *honeynet* com uma técnica baseada em pixels.

Daniel Keim [12] apresenta várias variações de técnicas baseadas em pixels, que envolvem a ordenação dos pixels ao longo do tempo usando linhas e diferentes curvas de preenchimento do espaço bidimensional (Peano-Hilbert, *Z-Curve*, etc.).

Técnicas hierárquicas particionam as múltiplas dimensões dos dados de forma hierárquica em subespaços que podem ser visualizados (em 2 ou 3 dimensões). Uma das técnicas hierárquicas mais conhecida é a *treemap* [19], que preenche uma área bidimensional com pequenos polígonos com área proporcional à importância para visualização. Outras características como cores ou texturas podem ser usadas para denotar informação adicional na visualização. Um exemplo de gráfico de *treemaps* é mostrado na Figura 4.22. Outra técnica hierárquica bem conhecida são os dendogramas, gráficos que particionam hierarquicamente um conjunto (relativamente pequeno) de dados, mostrando agrupamen-

tos e distâncias entre os dados representados. Um exemplo de dendograma é mostrado na Figura 4.14.

Técnicas baseadas em grafos servem para mostrar relações (arestas) entre objetos (vértices). A representação gráfica dos vértices e arestas pode ser usada para mostrar padrões e valores associados às relações (proximidade, intensidade, correlação, etc.) Várias técnicas de visualização usando grafos existem, e o estudo de algoritmos para posicionamento de vértices e arestas de forma a possibilitar a visualização de todo o conjunto é uma área de pesquisa bem ativa. Dois exemplos de visualização de grafos podem ser vistos nas Figuras **??** e 4.18.

Técnicas tridimensionais usam gráficos tridimensionais para visualização onde os dados são organizados em tipos de cenários, sendo muito mais efetivos na forma eletrônica (em *displays*) do que na forma impressa. Muitas destas técnicas são iterativas para que o usuário possa selecionar uma região ou subconjunto de dados adequado para visualização ou para que possa mudar o ponto de vista do cenário. Técnicas tridimensionais geralmente requerem razoável poder computacional ou *hardware* específico para uso efetivo, em especial para grandes volumes de dados.

Mapas são componentes de visualização universalmente conhecidos, e que podem ser usados como *background* para visualizações que tenham informações geográficas (frequentemente na forma de coordenadas pontuais ou de regiões geográficas). A Figura 4.5 mostra um exemplo de visualização de fenômenos geograficamente localizados: acessos a um conjunto de *honeypots* distribuídos no espaço da Internet brasileira, organizados por coordenadas aproximadas do IP de origem e com marcadores proporcionais ao número de acessos.

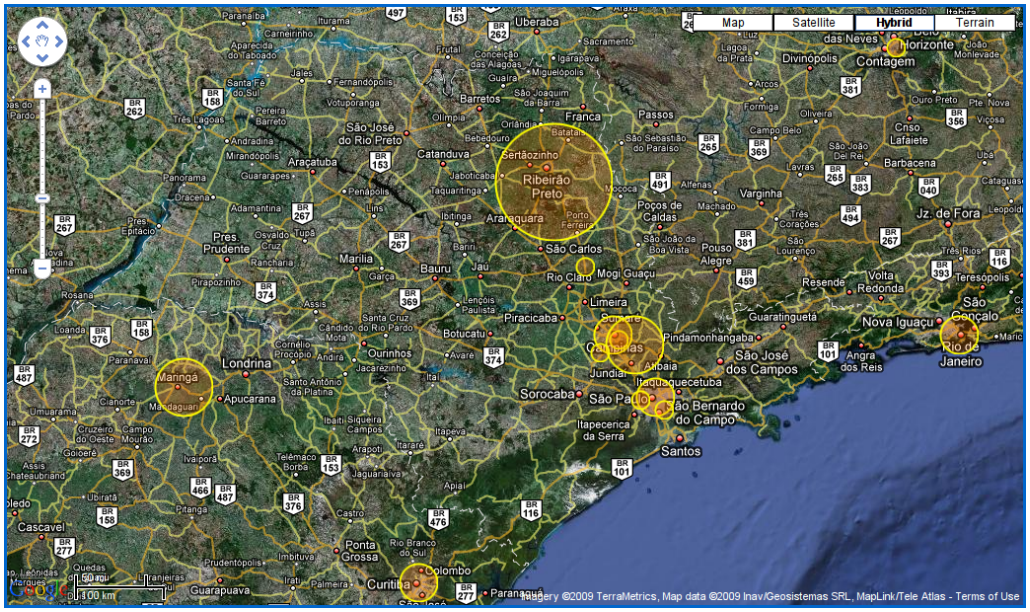

Figura 4.5. Número de acessos a uma *honeynet* (parte da região sudeste do Brasil)

Muitas outras técnicas de visualização existem, e várias não foram incluídas por questão de espaço. Vale a pena lembrar também que é possível e até comum misturar

características de diferentes técnicas de visualização ou usar diferentes técnicas de forma complementar.

4.3. Extração e Pré-processamento de Dados de Segurança para Visualização

Uma atividade fundamental para a segurança de sistemas é a monitoração. Monitorar e armazenar os registros dos eventos ocorridos em um sistema ou rede permitem uma análise posterior desses registros de forma a se procurar por ações suspeitas ou maliciosas, ou mesmo anomalias nos serviços do sistema. Os registros advindos da monitoração servem como dados de entrada para possibilitar a visualização de eventos de segurança e por isso um tratamento especial deve ser dispensado às fontes desses dados. Nesta seção veremos alguns tipos de dados de segurança e suas características, formas de pré-processamento de tais dados e como utilizá-los para visualização.

A necessidade da monitoração vem do fato que os ataques lançados contra sistemas computacionais deixam rastros, os quais podem ser utilizados para eventualmente identificar a parte invasora, ou para analisar como o ataque foi efetuado. Estes rastros correspondem às atividades realizadas em sistemas e redes alvo para que o ataque tenha sucesso, e armazenam informações diversificadas, tais como a data e a hora do ocorrido, o endereço IP de origem e de destino, serviços acessados/atacados, o conteúdo do fluxo de dados que resultou no comprometimento de uma rede de computadores ou de um sistema computacional, etc.

A interação resultante das atividades realizadas nos sistemas em geral é denominada de **evento**. Marty [15] define um evento como sendo *uma modificação ou situação observável ocorrida em um ambiente por um período de tempo determinado*. Ainda, que *um evento pode ser um estado específico ou uma mudança de estado de um sistema*.

O registro destes eventos é chamado de *log*, e é realizado através da coleta de dados de vários tipos de eventos de fontes diversificadas. Esta atividade é de vital importância para que o histórico dos eventos seja mantido e a fim de prover dados para a visualização destes. A seguir, o conceito de *log* será definido e discutiremos seus tipos, bem como os problemas de coleta, armazenamento e análise.

4.3.1. Fontes de dados: *Logs*

Antes de iniciarmos a apresentação dos tipos de fontes de dados, vamos definir o que é um *log*. Entende-se por *log* um registro de transação ou auditoria que consiste de um ou mais arquivos do tipo texto ou em formatos específicos gerados por certas aplicações, que permite a visualização dos eventos ocorridos em um sistema. Os *logs* são gerados por dispositivos computacionais relacionados aos componentes de um sistema, tais como aplicações, o sistema operacional, dispositivos de hardware, além da interação entre sistemas formando redes de computadores. Em linhas gerais, *logs* podem representar três tipos de eventos em um sistema: a atividade normal, alertas ou erro em algum dos componentes do sistema. Um *log* deve prover informação suficiente para que o evento possa ser identificado e compreendido.

Os dados ou mensagens de *log* podem variar conforme suas fontes geradoras, que vão desde o sistema operacional instalado, até a finalidade do sistema em si (servidor,

computador de usuário, roteador), não sendo limitados a um sistema específico. Algumas classes de sistemas ou dispositivos capazes de gerar *logs* são: dispositivos de rede, sistemas operacionais, aplicações, *firewalls*, sistemas de detecção de intrusão e antivírus.

Cada um destes tipos de *logs* tem o seu formato específico, definido pelo fabricante ou mantenedor do *software*/dispositivo gerador do *log*. A seguir serão mostrados exemplos de *logs* de alguns dos tipos citados, explicando as informações que os compõem.

4.3.1.1. *Logs* de Tráfego na Rede

Os *logs* do tráfego em uma rede, também conhecidos por *dumps*, contêm informações úteis para se identificar ocorrências relacionadas às conexões entre máquinas. Estas informações podem ser (mas não se limitam a) os endereços de rede das máquinas de origem e destino do tráfego, os serviços que serão executados durante a comunicação, os protocolos utilizados e os dados transferidos. Isto permite a reconstituição dos eventos de rede, podendo servir para se reconstruir partes de tráfego que antecederam um ataque, bem como o que ocorreu durante o mesmo (se foi bem-sucedido ou não, se houveram tentativas de conexão para outras redes com o objetivo de realizar varreduras, atacá-las ou obter ferramentas).

Um exemplo de *dump* pode ser observado adiante, que representa um pacote de início de conexão e contém muitas informações importantes para análise.

```
05:58:35.753776 00:21:29:d7:9a:ea > 00:19:e3:d3:d0:83, ethertype IPv4 (0x0800),
length 74: 10.1.1.91.443 > 192.168.1.100.60623: S 3658304570:3658304570(0) ack
3951756929 win 5792 <mss 1460,sackOK,timestamp 4044188756 742339032,nop,wscale 7>
```

Cada entrada no *log* de *dumps* aparece em uma linha. Os campos deste pacote são separados por espaços e são, na ordem em que aparecem:

- *timestamp* (marca o tempo em que o pacote foi capturado pela ferramenta);
- endereços Ethernet de origem e destino (MAC address)
- protocolo da camada de rede;
- tamanho do pacote (`length`);
- endereço IP e porta de origem > endereço IP e porta de destino;
- *flag* `SYN` habilitada (`S`);
- número de seqüência;
- *flag* `ACK` habilitada (`ack`);
- número de reconhecimento;
- tamanho da janela (`win`);
- tamanho máximo do segmento (`mss`) e outras opções.

As informações que podem ser extraídas de pacotes de rede são muitas, inclusive sobre o conteúdo destes pacotes, como por exemplo, sites acessados ou *logins* de usuários. Entretanto, utilizando apenas as informações de cabeçalho, como as mostradas acima, já é possível visualizar eventos simples e úteis, tais quais quantas conexões entram ou saem de uma determinada máquina, se um computador está gerando muito tráfego na rede e qual a localização das máquinas que estão atacando um determinado sistema.

4.3.1.2. *Logs* do Sistema Operacional

Estes *logs* contêm mensagens informativas sobre a inter-relação entre o sistema operacional, os componentes de hardware e os aplicativos em execução. Tentativas de acesso (local ou remoto), falhas de dispositivos e serviços ou mensagens do *kernel* do sistema são registradas neste tipo de *log*. Nos sistemas operacionais `unix-like` estes *logs* são armazenados no diretório `/var/log` e subdivididos por tipo de mecanismo monitorado ou informação a ser provida.

Para exemplificar um *log* de sistema, a seguir mostram-se eventos do *kernel* de um computador retirados de um *log* do sistema operacional, contendo a identificação do dispositivo de redes sem fio presente no computador citado.

```
Aug 20 07:08:40 Macintosh kernel[0]: AirPort: Link Down on en1
```

As informações que este tipo de *log* provê são:

- data e hora (*timestamp*);
- *hostname*, que é o identificador nominal do computador (a cargo do usuário ou *default*, nesse caso, Macintosh);
- agente gerador da mensagem (neste caso o *kernel* do sistema);
- serviço ou dispositivo que disparou este evento (no exemplo, o *driver* da interface de rede sem fio, AirPort);
- mensagem indicativa do evento.

4.3.1.3. *Logs* da Aplicação

Os *logs* das aplicações registram informações a respeito do funcionamento destas, tais como mensagens de início, finalização ou reinicialização de um serviço, acessos a recursos da aplicação e erros ocorridos. Muitas aplicações e ferramentas de segurança se utilizam das rotinas de geração de *logs* do próprio sistema operacional para registrar seus eventos.

Abaixo é mostrado um *log* da aplicação `ssh` contendo os mesmos campos de informação de um *log* do sistema operacional, uma vez que a aplicação se utilizou do mesmo mecanismo do sistema para registrar seus eventos. O primeiro evento corresponde

a um registro normal, indicando que o processo (*daemon*) está executando em estado de escuta na porta 22 local. O segundo e terceiro eventos correspondem a uma tentativa não autorizada de acesso, através de um ataque de força bruta por dicionário.

```
Oct  5 05:21:33 localhost sshd[2393]: Server listening on 0.0.0.0 port 22.
Oct  5 07:27:12 localhost sshd[3210]: Invalid user apple from X.Y.Z.132
Oct  5 07:27:12 localhost sshd[3210]: Failed password for invalid user apple from X.Y.Z.132 port 39427 ssh2
```

4.3.1.4. *Logs* de Sistema de Detecção de Intrusão

Os *logs* dos sistemas de detecção de intrusão (IDS, *Intrusion Detection Systems*) em geral assemelham-se a *dumps* do `tcpdump`, uma vez que alguns IDSs são baseados na biblioteca de captura de pacotes `libpcap`[4]. No caso do `Snort`[5], um sistema de detecção de intrusão baseado em assinaturas de ataque, junto com as informações do tráfego de rede há um alerta associado, correspondendo à identificação de uma possível tentativa de invasão. O identificador do alerta, seguido das informações contidas nos cabeçalhos dos protocolos da implementação da pilha TCP/IP podem ser verificados abaixo.

```
[**] [1:483:5] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
08/01-03:37:36.117652 10.10.10.2 -> 192.168.20.29
ICMP TTL:121 TOS:0x0 ID:49936 IpLen:20 DgmLen:92
Type:8  Code:0  ID:16009   Seq:3967  ECHO
[Xref => http://www.whitehats.com/info/IDS154]
```

4.3.1.5. Outros *logs*

Existem muitos outros tipos de *logs* voltados à descoberta de eventos de segurança, que tratam-se de *logs* de aplicações específicas ou *logs* de dispositivos de rede. Aplicações de segurança específicas são utilizadas para, por exemplo, monitorar a incidência de acessos via rede em sensores ou a incidência de ataques por *malware*. No caso dos dispositivos de rede, roteadores são configurados para exportar fluxos de dados, os quais podem prover informações interessantes sobre conjuntos de máquinas se comunicando, evidenciando atividades anômalas. A seguir, serão mostrados *logs* de sensores com as ferramentas *Honeyd*[6] e *Nepenthes*[7]. Na próxima seção serão expostos alguns gráficos utilizando os diferentes tipos de *logs* mencionados.

Honeyd

Honeyd é uma aplicação para implementação de *honeypots* – sistemas configurados especialmente para receber e monitorar ataques – de baixa interatividade. Com a utilização da ferramenta *Honeyd*, é possível instanciar centenas de endereços de Internet e emular sistemas operacionais e serviços de rede diversificados com apenas um computador [17].

---

[4] Maiores informações sobre o `tcpdump` e a `libpcap` podem ser obtidas em `http://www.tcpdump.org`.

[5] `http:www.snort.org`

[6] `http://www.honeyd.org`

[7] `http://nepenthes.carnivore.it`

Um exemplo de *log* provido pelo *Honeyd* está a seguir, no qual é mostrada uma conexão `telnet` estabelecida e tentativa de *login* falha (novamente formatado para adequação ao tamanho do texto):

```
Connection established: tcp (10.0.0.1:6324 - 192.168.1.1:23) <-> scripts/router-telnet.pl
E(10.0.0.1:6324 - 192.168.1.1:23): Attempted login: root/root123
```

Na ordem, as informações mostradas pelo *log* são:

- Estado da conexão;
- Protocolo;
- Par de endereços IP e portas comunicantes (origem e destino);
- Emulador utilizado (no exemplo, *script* que emula um terminal de `telnet` de roteador);
- Mensagem resultante da sessão (no exemplo, usuário e senha providos pelo atacante).

Nepenthes

Nepenthes é uma solução de *honeypot* de baixa interação desenvolvida para coletar *malware* de maneira automatizada [17]. A idéia principal por trás da ferramenta é a emulação de vulnerabilidades conhecidas em um serviço de rede, provendo o nível de interação com o serviço suficiente apenas para que este seja atacado por *malware* que se espalha automaticamente e o download de ferramentas para complementar o ataque seja feito, sem comprometer o sistema operacional base. Cada exemplar de *malware* coletado é armazenado em disco e todo o processo de obtenção do exemplar (tendo ou não sucesso) é registrado em *log*.

A partir dos *logs* do *Nepenthes*, pode-se extrair o caminho para repositórios de *malware* na Internet, como mostrado no *log* a seguir. Cada linha corresponde a um evento, e é composta pelo caminho para obtenção do *malware* e pelo MD5 do mesmo. Os endereços IP foram sanitizados para fins de anonimização, uma vez que muitos deles tratam-se de máquinas invadidas.

```
ftp://XX.YY.175.178:56172/lses.exe f63d2b8549208068b5912f2a4d5cfd03
link://XXX.YY.254.183:1449/tECs/A== de2a8e3f8e782d0a4847446d6bb39601
ftp://1:1@ZZZ.WWW.75.134:33606/svrsvc23.exe d33807a0843b40895766f85f277782f4
```

4.3.2. Problemas com a integridade das informações

Como visto, há uma vasta gama de dados diferentes providos por sistemas computacionais e que são utilizados na administração e segurança desses sistemas. Tais dados, por conter uma grande quantidade de informação, devem ser tratados com a finalidade de serem visualizados para prover rapidamente informações úteis sobre a segurança de uma rede ou sistema.

Entretanto, pode haver problemas na geração e armazenamento de *logs*, fazendo com que estes não sejam íntegros e gerem informações incorretas durante o processo de análise e visualização. Dentre os problemas que podem ocorrer quando da utilização de *logs* para visualização de segurança, os principais são:

- Dados incompletos;
- Erros de sincronização;
- Ruído nos *logs*.

4.3.2.1. Dados incompletos

O grande problema da incompletude dos dados é a possibilidade de interpretação errônea, tanto gerando alarmes falsos sobre eventos que são completamente normais, quanto deixando de dar a informação adequada no caso de eventos anômalos ou maliciosos.

Isto acontece quando os dados coletados – sejam eles quaisquer tipos de *logs* – sofrem uma parada abrupta no processo de coleta ou ocorre algum tipo de erro no armazenamento, causando corrupção nos arquivos. Na prática, isso pode acarretar não interpretação de um ou mais arquivos de um conjunto, tornando o resultado final incorreto.

Na Figura 4.6 é mostrado um gráfico de incidência de ataques de acordo com o protocolo de rede, no qual há espaços em branco, indicados por barras escuras. Tais espaços foram ocasionados por falhas de rede que impediram a máquina coletora de *logs* de recebê-los (e consequentemente armazená-los), modificando não só o gráfico, mas atrapalhando na geração das estatísticas para o referido período de tempo.

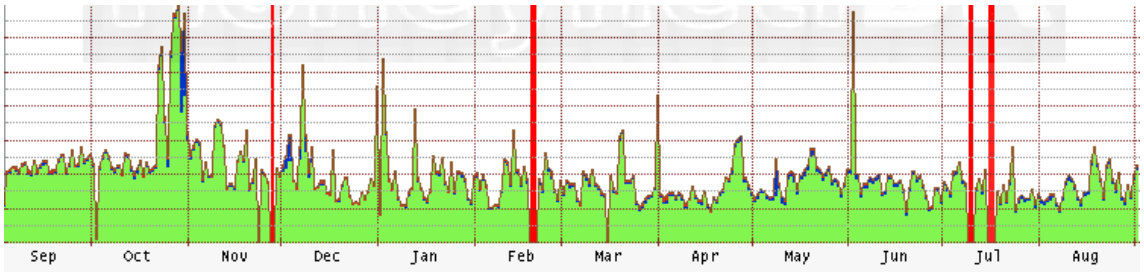

Figura 4.6. Gráfico de informações de acessos à rede de um ano, com espaços faltantes

No caso de análises para identificar intrusões, é necessário o correlacionamento de diferentes fontes de dados para que se chegue a uma conclusão acerca da extensão de um ataque efetuado. Por exemplo, uma invasão a um servidor de uma rede pode gerar *logs* no *gateway* da rede, no *firewall*, no sistema de detecção de intrusão, no sistema operacional do servidor e na aplicação atacada. A intersecção destas diversas fontes de dados é que irá gerar as informações necessárias para se caracterizar como o ataque foi realizado, que outro sistema foi atacado e qual foi a extensão do dano sofrido. Dado faltantes ou corrompidos certamente prejudicam o andamento da investigação e das conclusões, uma vez que pode não ser possível visualizar adequadamente todos os eventos que ocorreram.

Para garantir que isso não ocorra, deve-se investir principalmente no bom projeto da arquitetura que irá efetuar a coleta e armazenamento dos *logs*, levando em conta a disponibilidade, integridade e monitoramento constante dos processos envolvidos.

4.3.2.2. Erros de sincronização

Ainda que a arquitetura de coleta e armazenamento de *logs* seja adequada, pode ocorrer um outro problema, igualmente prejudicial ao dos dados incompletos: a falta de sincronismo entre os relógios dos dispositivos componentes do sistema. É importante que todos os componentes (servidores, roteadores, máquinas de usuários) estejam com o mesmo horário, ou indicando corretamente as *time zones*. Erros na sincronização dos relógios podem invalidar completamente uma investigação e não permitir a visualização correta das informações, pois os eventos de um ataque devem estar necessariamente em ordem cronológica.

Mesmo que os eventos possam ser colocados na ordem correta, é importante levar em conta a precisão de tempo dos registros. Marty escreve em *Applied Security Visualization* [15] que *transações financeiras são extremamente sensíveis no que diz respeito ao tempo, e diferenças de milisegundos podem fazer uma diferença significativa em como eles são interpretados ou os efeitos que eles têm*.

4.3.2.3. Ruído nos *logs*

Um outro problema muito comum é o ruído, o qual ocorre quando eventos que não deveriam ser registrados aparecem nos *logs*, atrapalhando a visualização dos eventos importantes ou ocasionando erros na interpretação dos resultados.

Os ruídos ocorrem quando não é feita a filtragem adequada no pré-processamento dos dados, deixando aparecer por exemplo, no caso de *logs* de rede, os acessos administrativos ou tráfego comum de protocolos de roteamento. No caso da visualização de eventos do sistema operacional, como a análise dinâmica da execução de um código malicioso, o ruído pode ser exemplificado pelo aparecimento de chamadas de sistema das ferramentas usadas para análise junto com as chamadas do programa malicioso.

Para resolução de problemas de ruído, é necessário o tratamento meticuloso dos dados que irão gerar um *log*, atentando para a retirada das informações normais, comuns ou sem utilidade para a detecção do evento.

4.4. Técnicas de Visualização com Aplicações a Dados de Segurança

Utilizando a diversidade de *logs* gerados por sistemas operacionais, dispositivos de rede e mecanismos para segurança, podemos visualizar os eventos de maneira mais prática do que simplesmente a leitura de um arquivo com informações textuais. Técnicas de visualização de eventos de segurança utilizando gráficos permitem a identificação rápida de atividades suspeitas ocorrendo em uma rede ou sistema, auxiliando na resposta ao incidente.

Nesta seção mostraremos várias formas de se visualizar eventos de segurança que

são utilizadas pelos autores em suas redes institucionais e outros exemplos da área.

4.4.1. Dados geográficos de atividades maliciosas

Na seção anterior foi mostrado que um *log* provido por um *honeypot* utilizando *Honeyd* contém informações sobre um endereço IP utilizado em um ataque, bem como o serviço atacado, o protocolo da conexão, entre outras informações.

O Consórcio Brasileiro de Honeypots [8] criou há alguns anos o *Projeto Honeypots Distribuídos* para aumentar a detecção de incidentes, correlacionamento de eventos de segurança e, principalmente, observação das tendências no que diz respeito aos ataques ocorrendo no ciberespaço brasileiro. Este Projeto trata-se da implantação de uma rede de *honeypots* com a ferramenta *Honeyd* instalada, espalhada por mais de 20 instituições (públicas e privadas) em todo o território nacional. Além disso, os dados são analisados e sumários com as informações diárias coletadas são enviados aos membros, para que esses possam averiguar quais tipos de acesso seu sensor está recebendo.

A partir desse dados, são gerados gráficos para identificação dos países que foram origens de ataques à *honeypots* do Consórcio. Os gráficos são gerados de hora em hora, e ficam disponíveis para consulta[9]. Os dados de entrada para a construção destes gráficossão obtidos por meio do *parsing* dos *logs* dos sensores com *Honeyd* do Consórcio e tratamento destes a fim de pontuar quantos acessos um determinado endereço IP efetuou, e de onde tal endereço é proveniente.

A visualização de como os ataques provenientes de diversos países aos sensores nacionais evoluem, de hora em hora, pode ser observada por este tipo de gráfico. A Figura 4.7 ilustra os ataques ocorridos no período de uma hora.

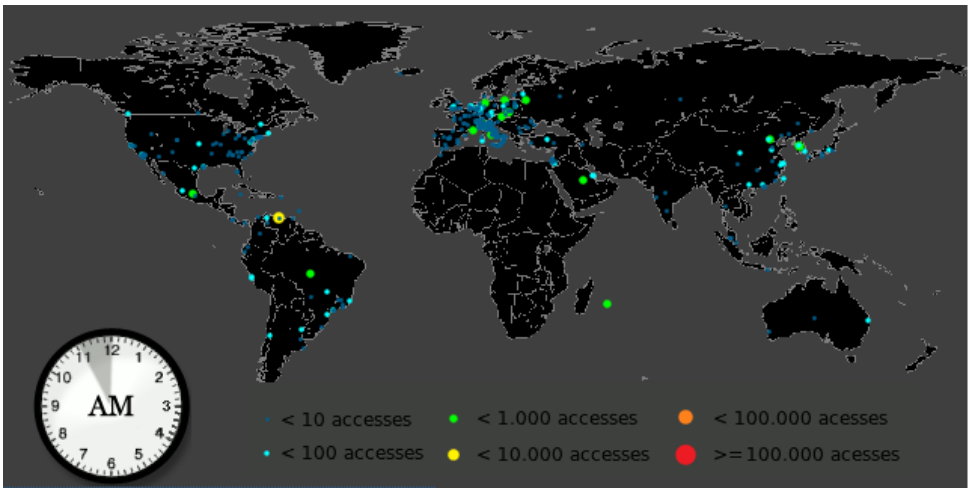

Figura 4.7. Evolução de ataques a *honeypots* do Consórcio Brasileiro de Honeypots por hora

Utilizando técnica similar, os *logs* da ferramenta *Nepenthes* podem ser tratados para extrair os endereços IP utilizados para o download de *malware*, gerando assim um

[8] http://www.honeypots-alliance.org.br
[9] http://www.dssi.cti.gov.br/dssi/statistics.html

mapa dos repositórios de *malware* ao redor do mundo, ou ainda, relacionar quais países estão provendo quais amostras de *malware*, mapeando assim sua distribuição.

Outra iniciativa de monitoração de atividades maliciosas utilizando visualização geográfica é implementada por *The Shadowserver Foundation*[10]. Os endereços IP de servidores de controle de *botnets* monitoradas são convertidos em coordenadas geográficas e ilustrados em um mapa do globo, seguindo a mesma idéia do mapa apresentado anteriormente. Mais servidores de controle em uma mesma posição geram pontos de diâmetro maior no mapa.

4.4.2. Visualização de Campanhas de Spam

O CERT.br[11] instanciou, em 2007, o *Projeto SpamPots*[12] para levantar dados sobre o comprometimento de máquinas conectadas à Internet com a finalidade de enviar *spam*. Em parceria com o Departamento de Ciência da Computação da UFMG[13], os dados coletados de *spam* foram analisados em busca de comportamentos que identificassem o padrão de atuação dos *spammers*, utilizando técnicas de mineração de dados. A identificação desses padrões envolve a separação de uma grande massa de *spams* em campanhas, isto é, conjuntos de mensagens que compartilham características frequentes e possuem poucas características infrequentes as quais servem para ofuscar o real objetivo da mensagem. Na Figura 4.8, um exemplo de campanha de *spam* é mostrado, onde cada atributo extraído de uma mensagem de *spam* – idioma, *layout*, *tokens* das URLs contidas – é colocado em uma estrutura de árvore de padrões frequentes e representado por uma cor diferente. Este tipo de gráfico permite a visualização dos padrões para criação de uma campanha e consequente melhor entendimento das técnicas utilizadas por um *spammer* para ofuscar as mensagens e driblar os filtros *antispam* [3].

4.4.3. Atuação de *malware*

Tem sido dada muita atenção a análise de software malicioso (*malware*), que é a maior ameaça aos sistemas computacionais atualmente. A análise de comportamento dos binários pode fornecer informações sobre a similaridade entre diferente exemplares de *malware* pertencentes a um mesmo grupo.

O levantamento das chamadas de sistema (*syscalls*) do *malware* no sistema operacional, isto é, as ações executadas que indicam o comportamento malicioso, podem servir de base para o agrupamento dos *malware* em famílias. É possível também identificar, através da análise comportamental, binários que representam exatamente o mesmo *malware*.

Um trabalho em andamento que está sendo feito pelos autores é a categorização de *malware* de acordo com as chamadas de sistema realizada. Na Figura 4.9 há um exemplo com uma pequena amostra de visualização de exemplares malware através das *syscalls* efetuadas. Quanto mais escura a linha, maior a incidência de *syscalls* de um determinado tipo, onde cada linha denota uma *syscall* diferente. O programa de visualização,

[10] `http://www.shadowserver.org`
[11] `http://www.cert.br`
[12] `http://www.cert.br/docs/whitepapers/spampots`
[13] `http://www.dcc.ufmg.br`

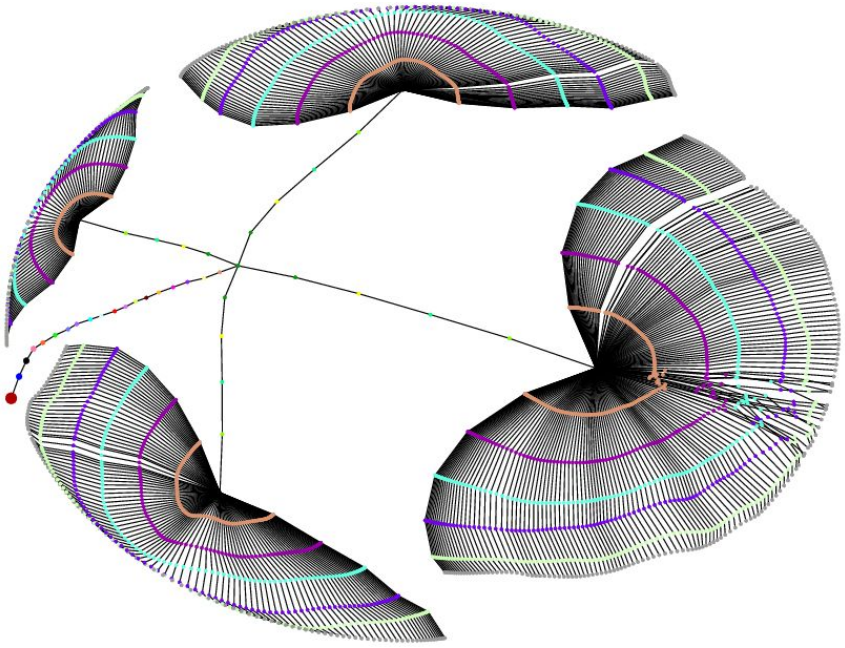

Figura 4.8. Exemplo de árvore de padrões para campanha de *spam*

construído em Java, obtém arquivos com as *syscalls* dos *malware* e desenha os gráficos observados na figura. Caso não se consiga obter o comportamento do exemplar, o gráfico aparece em branco. No caso de não existir um arquivo com as *syscalls*, ou se este estiver corrompido, o gráfico do respectivo exemplar é marcado com um **X** em vermelho. Na Figura, pode-se notar a similaridade de comportamento entre dois exemplares distintos identificados como **Trojan.KillAV**, apenas pela observação dos dois gráficos localizados na porção superior da Figura.

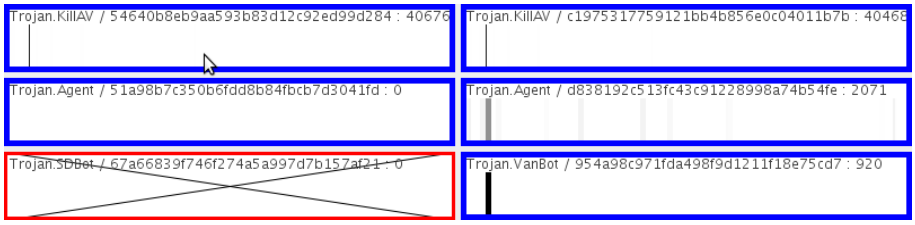

Figura 4.9. Visualização de comportamento de *malware* através das *syscalls* efetuadas

4.4.4. Sensores de segurança

Ainda utilizando dados do *Honeyd*, é possível gerar gráficos simples, apenas para se ter estatísticas de serviços mais acessados ou protocolos mais utilizados em ataques. Através do gráfico mostrado na Figura 4.10, temos uma estatística diária sobre quais portas são mais acessadas e o protocolo utilizado, o que é bastante útil para a identificação dos serviços mais atacados a fim de mantê-los atualizados e verificar se estão sendo implementadas medidas de proteção adequadas.

Outro gráfico simples, este servindo para a observação de tendências ao longo do tempo pode ser visto na Figura 4.11, extraída da página do *ShadowServer*.

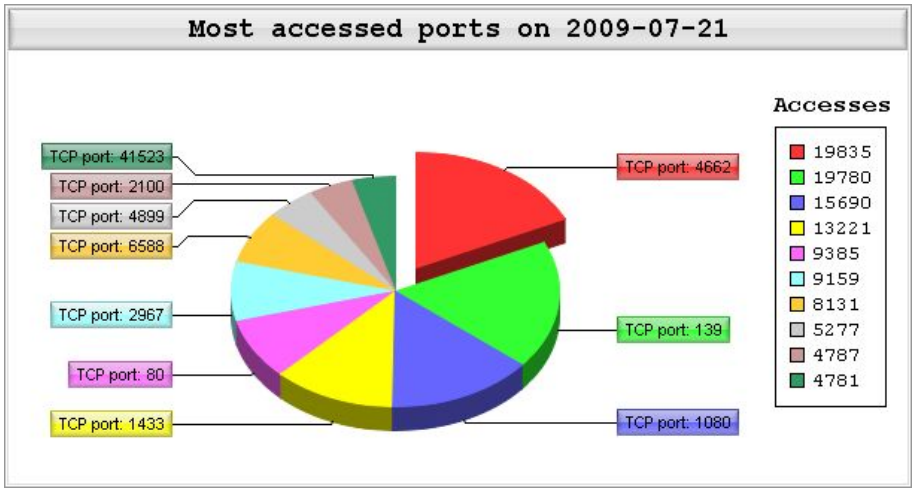

Figura 4.10. Portas mais acessadas em um *honeypot* durante o período de um dia

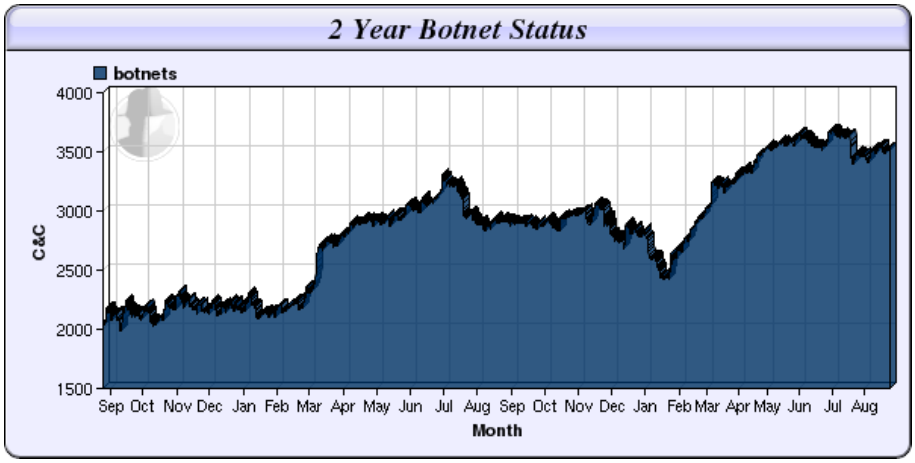

Figura 4.11. Quantidade de servidores de controle de *botnets* monitorados ao longo do tempo

4.4.5. Outros Exemplos

A seguir, serão mostrados alguns exemplos comentados de aplicação de técnicas de visualização em segurança de redes, sistemas e dados, selecionadas da literatura técnica recente.

Conti et al. [5] apresentam cenários onde é necessário analisar (ou modificar) arquivos de forma independente de contexto (isto é, sem conhecimento explícito sobre o formato real e função dos arquivos). Alguns destes cenários são baseados em problemas relacionados com análise forense, análise de *malware*, identificação e auditoria em arquivos, entre outros. Baseado em alguns destes cenários os autores implementam técnicas complementares de visualização de um mesmo arquivo, em uma ferramenta que apresenta informações textuais junto com informações binárias, apresentadas como imagens (Figura 4.12). Os autores demonstram como as técnicas implementadas podem ser usadas para rapida e facilmente identificar texto em arquivos binários, regularidade em arquivos com registros de tamanho fixo, mudança em estruturas de arquivos e outros

exemplos.

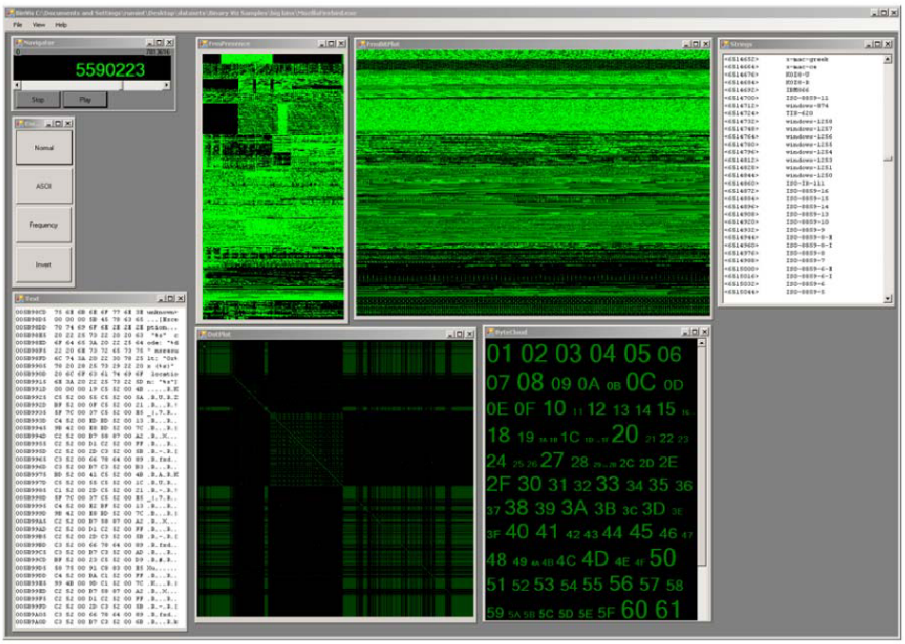

Figura 4.12. Interface da aplicação desenvolvida por Conti et al. [5].

Fischer et al. [6] mostram técnicas de análise e visualização de dados coletados do protocolo NetFlow. Uma variante da técnica *treemap* é usada para indicar conexões entre máquinas externas e internas à rede sendo monitorada. Como exemplo de visualização do sistema proposto, a Figura 4.13 mostra de forma gráfica as conexões feitas por uma *botnet* com 120 máquinas para as máquinas da Universidade de Konstanz (Alemanha) durante um ataque em 2008. De acordo com os autores o ataque, por ter sido feito de forma distribuída, não foi severo o suficiente para deteção por sistemas existentes na universidade na época.

Karim et al. [11] descrevem métodos para a construção de modelos de filogenia de *malware* para estudar melhor a evolução do código do *malware*. Embora a abordagem dos autores não seja diretamente relacionada com visualização, eles usam em seu artigo uma técnica composta de dendograma com anotações que pode ser replicada e adaptada para outras aplicações de visualização de dados representáveis de forma hierárquica. A Figura 4.14 mostra a técnica usada pelos autores.

Bilar [2] analisa a estrutura de chamada de funções (*callgraphs*) de *malware* e de aplicações legítimas quanto à algumas propriedades: número de chamadas a funções internas (normais), número de chamadas a funções externas de forma estática (bibliotecas)ou dinâmica (*imports*) e número de *thunks* (chamadas curtas, geralmente envelopes de funções).

Os resultados são apresentados de forma estatística e com vários gráficos de espalhamento que mostram, para os conjuntos de *malware* e de aplicações legítimas, as distribuições das variáveis estudadas, duas a duas (o que seria um bom caso para uso de uma matriz de gráficos de espalhamento). A cor de cada ponto é proporcional ao tamanho

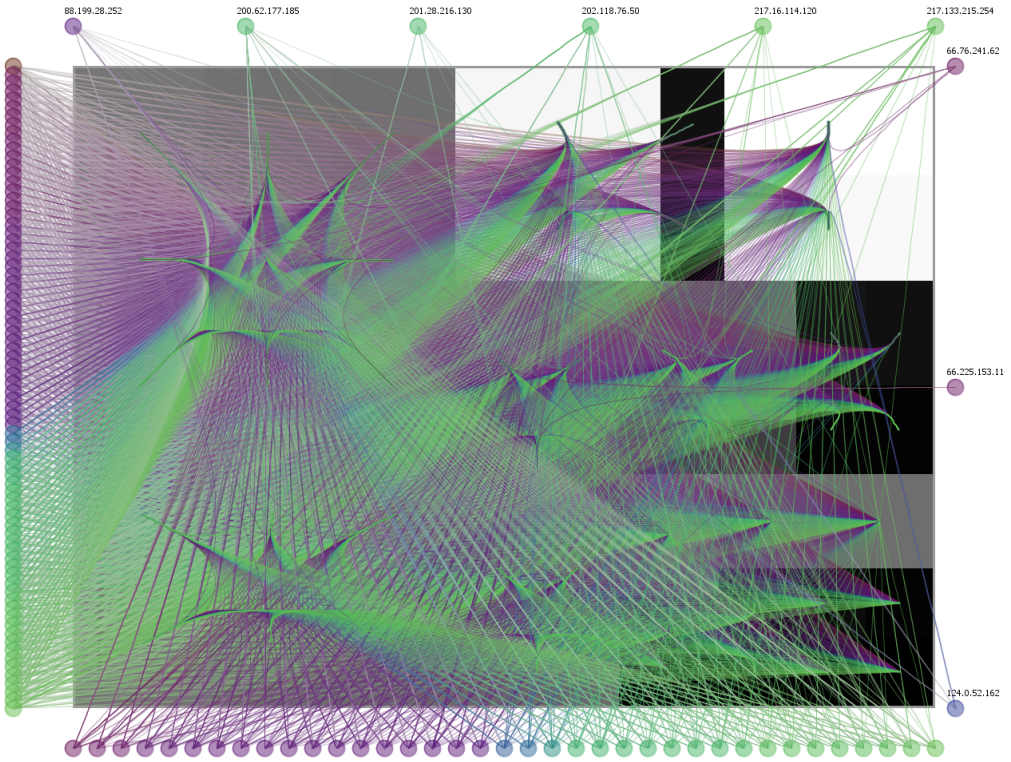

Figura 4.13. Visualização das conexões feitas durante um ataque de uma *botnet*, usada por Fischer et al. [6].

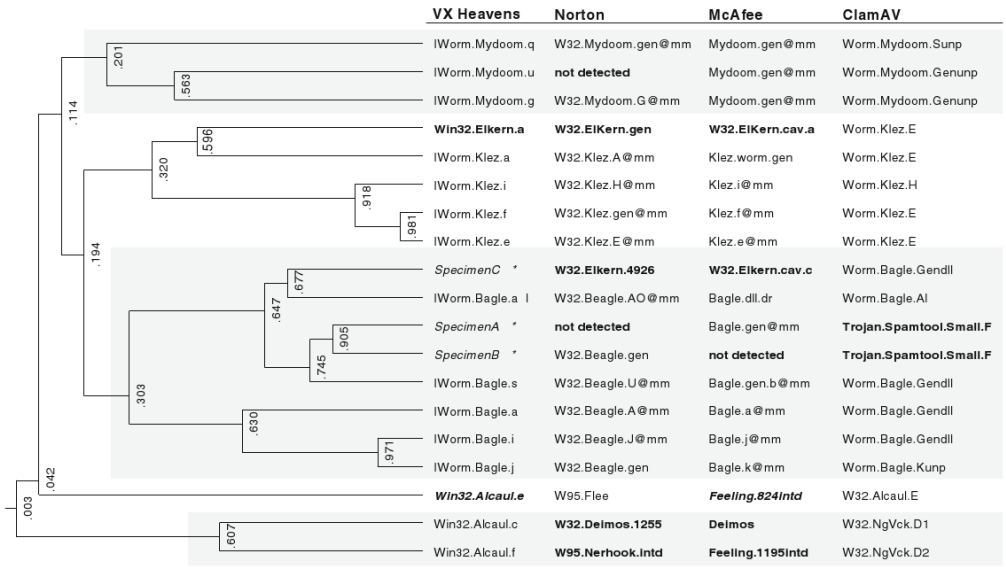

Figura 4.14. Dendograma mostrando um modelo de filogenia de *malware* e respectiva classificação por alguns anti-vírus (Karim et al. [11].

do executável. O exemplo de visualização dos dados dos *callgraphs* criado por Bilar [2] é mostrado na Figura 4.15.

Onut et al. [16] apresenta uma técnica para auxiliar a deteção de tentativas de in-

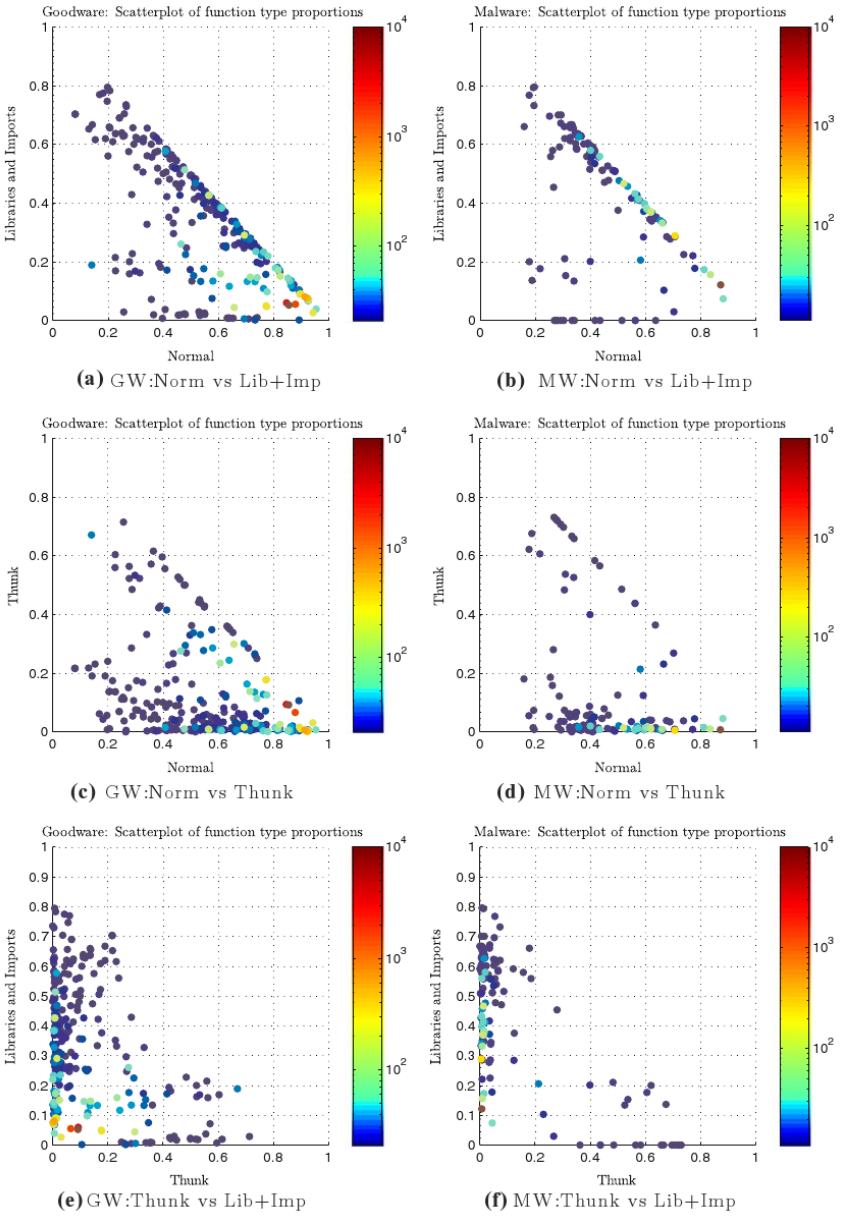

Figura 4.15. Gráficos de espalhamento de métricas de *callgraphs* de *malware* e de software legítimo (Bilar [2]).

trusão através da visualização da rede sendo estudada como uma comunidade de sistemas que trocam mensagens entre si. A técnica possibilita visualizar a quantidade de troca de mensagens entre vários serviços simultaneamente.

Wang e Zhang [21] usam uma técnica semelhante à de coordenadas paralelas para visualizar, em três dimensões, datas, tipos, tamanhos, origem e destino de e-mails enviados para uma rede. A técnica permite visualizar alguns comportamentos conhecidos e facilmente interpretáveis e também comportamentos inusitados. A Figura 4.16 mostra algumas instâncias de *relay* de *spam* pelo servidor dos autores.

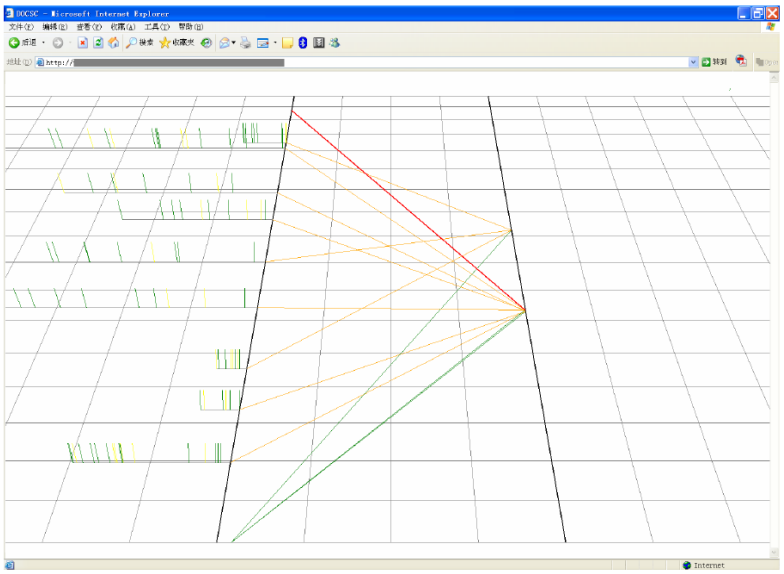

Figura 4.16. Visualização de tráfego de e-mail mostrando contas usadas para envio de *spam* [21].

4.5. Sugestões para Estudos Complementares

4.5.1. Algumas ferramentas e APIs para visualização

Nesta subseção apresentaremos algumas ferramentas para visualização de dados em geral, mas que podem ser usadas para visualização de dados relacionado a segurança. Existem muitas ferramentas e APIs (*Application Programming Interface*, conjunto de funções que podem ser usadas com uma linguagem de programação para habilitar esta a realizar tarefas específicas), nesta subseção serão listadas algumas mais interessantes e que são abertas e/ou gratuitas.

Uma das ferramentas que oferece mais praticidade para visualização de dados de segurança atualmente é o Projeto DAVIX [14]. Trata-se de um *live CD* com diversas ferramentas livres para visualização e análise de dados, algumas das quais discutidas separadamente mais adiante neste texto. Na Figura 4.17, é mostrado um *snapshot* da tela do DAVIX com várias ferramentas sendo aplicadas para visualização de tráfego de rede capturado.

JUNG (*Java Universal Network/Graph Framework*)[15] é uma API para processamento de grafos (conjuntos de vértices e arestas que conectam estes vértices). A API permite a representação de grafos de forma simples e contém algoritmos de teoria dos grafos, mineração de dados e de análises de redes sociais para criação de agrupamentos, decomposição de grafos, análise estatística, cálculo de métricas e fluxos, etc.

A Figura 4.18 mostra uma visualização simples de relações entre diferentes *malware* criadas usando a API JUNG. Foram selecionados aproximadamente 200 *malware* a partir de um conjunto de mais de 22.000, provenientes de base de dados de amostras coletadas pelos autores em *honeynets*, *honeypots*, repositórios na Internet e anexos de *e-mail*.

[14] `http://www.secviz.org/node/89`
[15] `http://jung.sourceforge.net/`

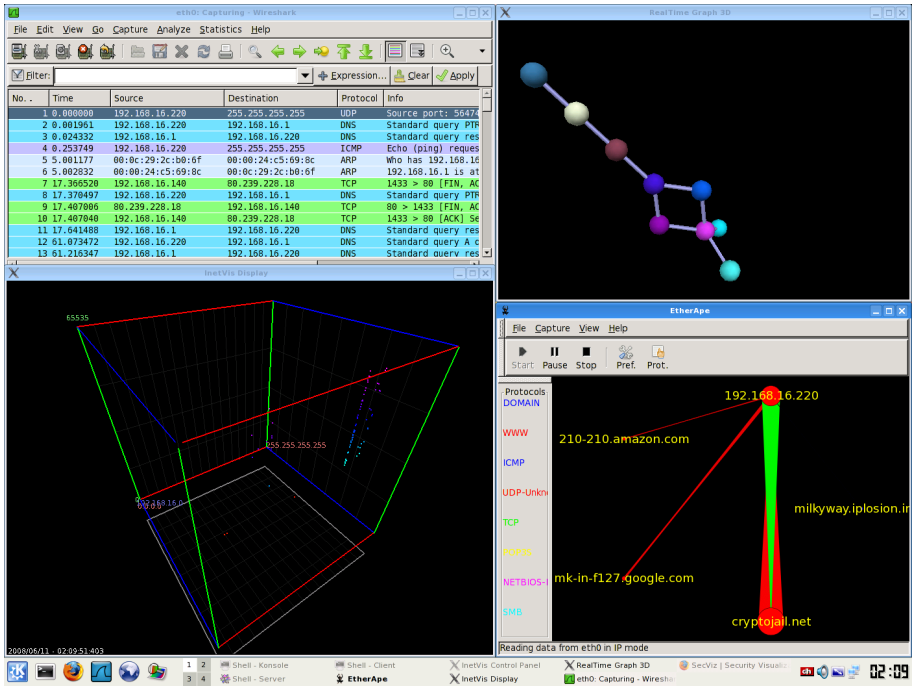

Figura 4.17. DAVIX em uso, com ferramentas gerando gráficos para visualização de tráfego de rede

O espaço amostral de *malware* foi enviado para o *VirusTotal*[16] e apenas os resultados positivos de identificação foram armazenados e associados ao *malware*. Os vértices dos grafos são os descritores dos *malware*, e as arestas entre os descritores tem um peso associado, calculado como o grau de concordância entre anti-vírus para os *malware* (para dois *malware* contamos quantas vezes os anti-vírus identificam eles como sendo da mesma família).

Para aproveitar melhor as capacidades da API alguns vértices foram coloridos de forma diferente: vértices correspondentes a *malware* que sugerem que o mesmo faça parte da família de *backdoors* Tsunami[17] são pintados de vermelho para diferenciação. A Figura 4.18 mostra o grafo com os vértices em um *layout* que os agrupa dependendo do valor associado a suas arestas, criando grupos ou *clusters* visíveis – todos os *malware* identificados como da família Tsunami aparecem em um grupo isolado, exceto por dois. Uma versão dinâmica desta ferramenta de visualização está em desenvolvimento pelos autores, e possibilitará a identificação de *malware* que são classificados supervisionadamente em uma categoria mas apresentam características de outras categorias.

Existem outras alternativas para visualização de grafos, com características diferentes. A API **SUVI**[18], que tem mais opções para visualização mas que por outro lado não contém algoritmos para processamento dos grafos, e não é mantida há anos, embora esteja aparentemente estável. **LGL** (*Large Graph Layout*)[19] é uma coleção de aplicativos que permite a visualização de grafos com grandes quantidades de arestas e vértices, e é

[16] `http://www.virustotal.com`
[17] `http://www.viruslist.com/en/viruses/encyclopedia?virusid=47843`
[18] `http://suvi.sourceforge.net/`
[19] `http://lgl.sourceforge.net/`

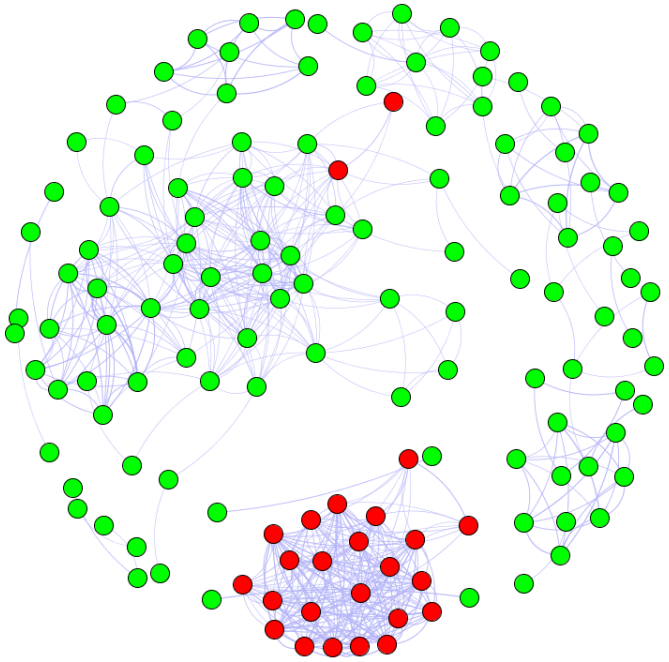

Figura 4.18. Visualização de similaridade entre *malware* com a API JUNG (com agrupamentos).

usado para visualização de dados biológicos e da estrutura da Internet[20]. **GVF** (*Graph Visualization Framework*)[21] é um conjunto de funções e aplicações que também permite a representação e desenho de grafos, com ferramentas interativas que permitem a criação de visualizações complexas (com *layouts* diferentes para subgrafos em um mesmo grafo).

Ainda outra ferramenta para visualização de grafos é **Graphviz** (*Graph Visualization Software*)[22], que usa formatos de texto para representar os grafos e que também oferece diferentes possibilidades de *layout*. A Figura 4.19 mostra um grafo que representa 300 *sites* em mais de 40 países, que pode ser usado para identificar incidentes na rede e para dar suporte à manutenção da mesma (dados obtidos da galeria de aplicações do Graphviz em `http://www.graphviz.org/Gallery/twopi/twopi2.html`).

O software **Tulip**[23] também permite a visualização e processamento de dados em forma de grafos. Em particular possui algoritmos para clusterização ou agrupamento de grafos, permitindo a navegação nos *clusters* formados. A Figura 4.20 mostra um exemplo de grafo visualizado com o Tulip. O gráfico contém a estrutura de um servidor HTTP.

OpenDX[24] é um ambiente de visualização de dados científicos, originalmente desenvolvido pela IBM e posteriormente disponibilizado como código aberto. OpenDX apresenta uma interface de programação visual onde operadores são representados grafi-

[20] `http://www.opte.org/`
[21] `http://gvf.sourceforge.net/`
[22] `http://www.graphviz.org/`
[23] `http://tulip-software.org/`
[24] `http://www.opendx.org/`

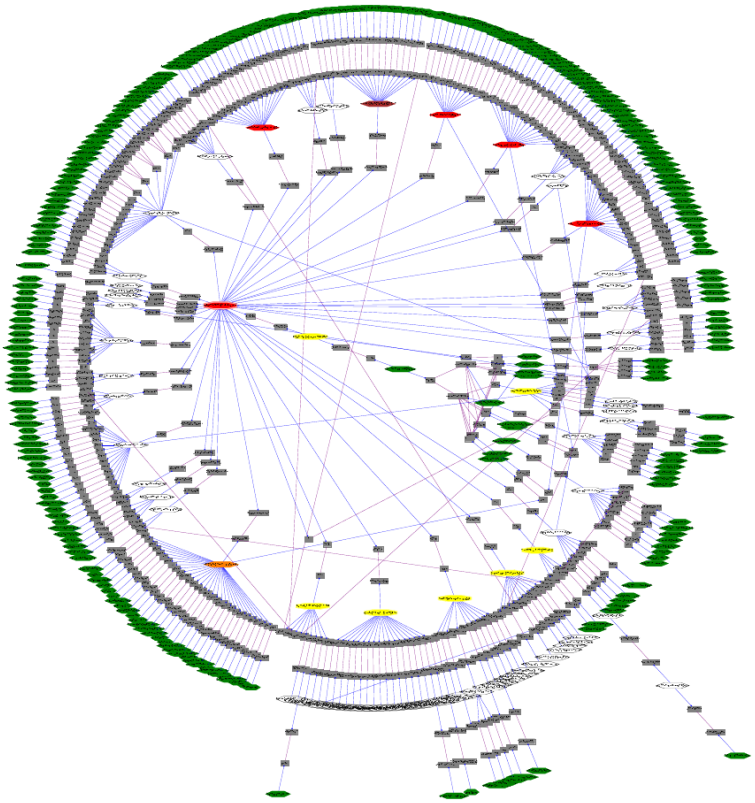

Figura 4.19. Visualização da estrutura de uma rede de computadores com Graphviz.

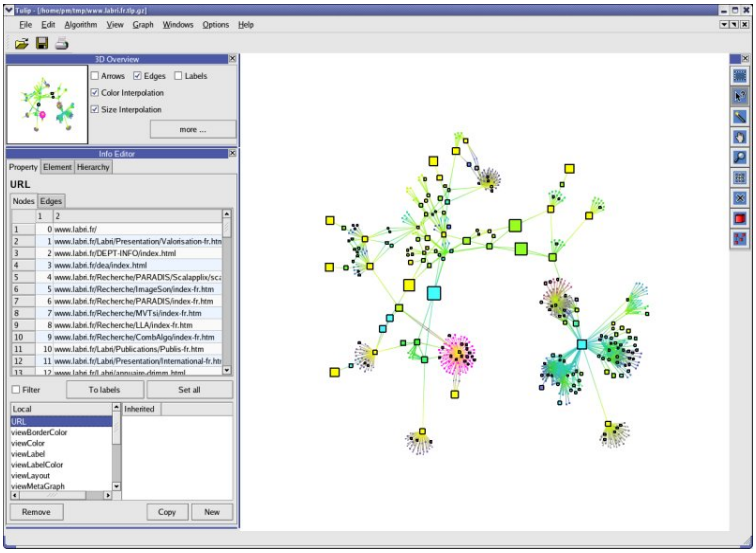

Figura 4.20. Visualização de estrutura de arquivos em um servidor WWW (criada pelo software Tulip)

camente e fluxos de comando interligam estes operadores, contendo também bibliotecas de funções que podem ser chamadas a partir de aplicações escritas em C, Python ou Tcl/TK.

The InfoVis Toolkit[25] é uma outra API para criação de visualizações (através de programação em Java usando as classes da API) que usa uma estrutura de dados simples (tabela) para representar os dados a ser visualizados. A API inclui nove tipos diferentes de gráficos, que podem ser customizados. Esta API usa uma outra API de renderização de gráficos (Agile2D) que pode aumentar consideravelmente a velocidade de renderização, possibilitando a criação de gráficos que usam muitos dados. A Figura 4.21 mostra um exemplo de aplicação desenvolvido com esta API.

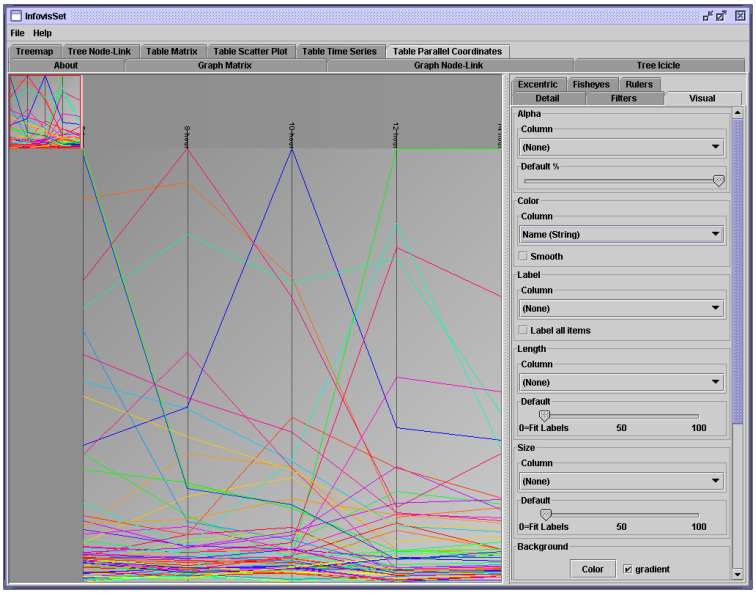

Figura 4.21. Gráfico de coordenadas paralelas criado pelo Infovis Toolkit.

Processing é um ambiente de *design*, prototipagem gráfica e visualização, baseado em programação, com características da linguagem Java. O desenvolvimento de aplicações em Processing é feito em uma IDE (*Integrated Development Environment*, ambiente integrado de desenvolvimento) que simplifica bastante a edição dos comandos, inclusão de dados auxiliares e criação de aplicações e *applets* em Java prontas para execução em qualquer ambiente computacional que tenha uma máquina virtual Java recente. Inicialmente usada como ferramenta de *design* rápido, Processing tem sido usada como ferramenta para desenvolvimento de aplicações de visualização de dados de diversos tipos [7]. Uma boa introdução a Processing para pessoas com pouca experiência em programação é o livro de Daniel Shiffman [18].

Além da simplicidade para desenvolvimento e empacotamento de aplicativos prontos para execução, Processing facilita a criação de documentos em diversos formatos gráficos, inclusive PDF (*Portable Document Format*) e SVG (*Scalable Vector Graphics*). É possível incluir bibliotecas externas para finalidades específicas.

Como exemplo de visualização usando Processing podemos ver a Figura 4.22, que usa uma biblioteca de geração de *treemaps* (seção 4.2.1) e uma pequena aplicação em Processing (baseada em um exemplo em [7]) para mostrar frequências de palavras em uma base de dados com mensagens possivelmente relacionadas a fraudes, obtida do

[25] `http://ivtk.sourceforge.net/`

grupo *Computational Linguistics And Information Retrieval (CLAIR)*[26] na Universidade do Michigan). A Figura 4.22 mostra o gráfico do tipo *treemap* com a aplicação de um filtro primitivo de palavras: somente as palavras com seis ou mais caracteres foram usadas para a geração do gráfico. Neste tipo de gráfico, cada área é proporcional à ocorrência daquela palavra na base de dados. Pela figura podemos observar a frequência relativamente alta de ocorrência de palavras associadas a este tipo de fraude.

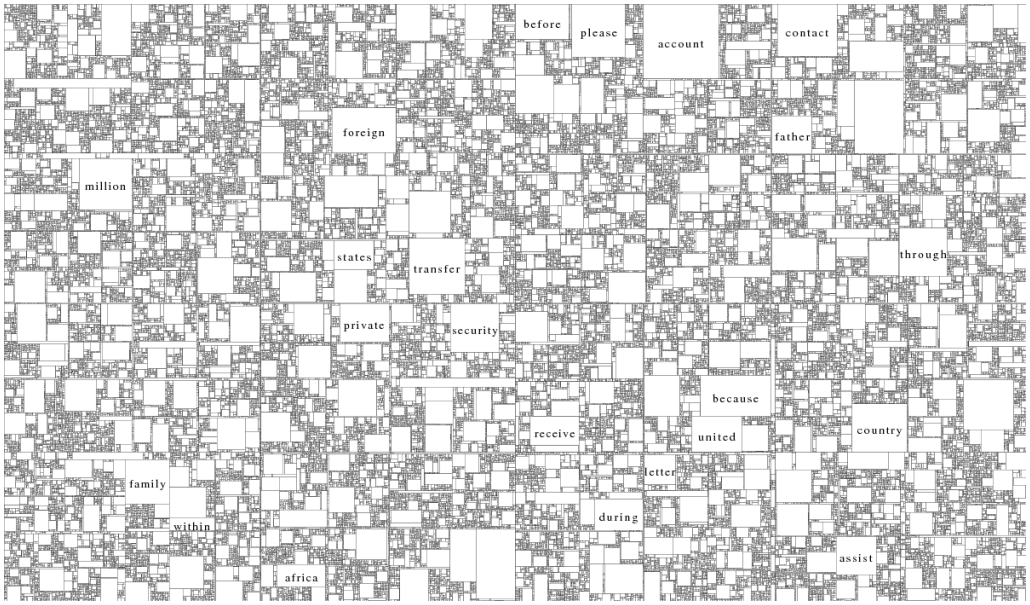

Figura 4.22. *Treemap* mostrando frequência relativa de palavras em base de dados (filtrada) de mensagens relacionadas a fraudes.

Prefuse[27] é uma API para a linguagem Java que permite a representação e visualização de dados de diversos tipos. A API também tem classes e métodos para interatividade com a representação visual dos dados, animação, busca textual e por comandos com sintaxe semelhante à de SQL (*Structured Query Language*) e outras funções. Uma versão alternativa de Prefuse (Prefuse Flare) pode ser usada em aplicações desenvolvidas em Flash. **Piccolo2D**[28] é outra API para as linguagens Java e C# que também tenta liberar o programador de se envolver com detalhes de programação de gráficos, fornecendo classes que correspondem a tarefas específicas de visualização. Uma das características mais interessantes de Piccolo2D é a possibilidade de uso de *Zoomable User Interfaces*, interfaces que permitem o uso de muitos elementos gráficos em uma pequena área (monitor do computador), com controles para que o usuário da aplicação possa selecionar e ampliar trechos dos gráficos.

Improvise[29] é uma ferramenta para *design* de visualizações interativas que usa o conceito de visões múltiplas coordenadas. Usuários desta aplicação podem construir relações visuais entre dados e interagir com estas relações de várias formas, inclusive usando uma linguagem visual. Improvise não é distribuída como uma API, mas como o

[26] `http://tangra.si.umich.edu/clair/`
[27] `http://www.prefuse.org/`
[28] `http://www.piccolo2d.org/`
[29] `http://www.cs.ou.edu/~weaver/improvise/index.html`

código-fonte está disponível para *download*, programadores com experiência podem usar alguns de seus componentes em aplicações específicas.

Outras ferramentas abertas como Octave[30], Scilab[31], R[32], Weka[33], RapidMiner[34] tem módulos para análise e visualização de dados. Várias destas podem também ter os módulos incluidos em aplicações desenvolvidas pelo usuário, e para alguns podemos desenvolver novos módulos através de programação em linguagens genéricas ou específicas.

Referências

[1] J. Beddow. Shape Coding of Multidimensional Data on a Mircocomputer Display. In *Visualization '90. Proceedings of the First IEEE Conference on Visualization*, pages 238–246, 1990.

[2] Daniel Bilar. On callgraphs and generative mechanisms. *Journal of Computer Virology*, 3(4):163–186, 2007.

[3] Pedro H. Calais, Douglas E. V. Pires, Dorgival Olavo Guedes, Wagner Meira Jr, Cristine Hoepers, and Klaus Steding-Jessen. A campaign-based characterization of spamming strategies. In *Proceedings of Fifth Conference on E-mail and Anti-Spam - CEAS*, 2008.

[4] W. S. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[5] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. Visual Reverse Engineering of Binary and Data Files. In John R. Goodall, Gregory Conti, and Kwan-Liu Ma, editors, *Visualization for Computer Security – 5th International Workshop, VizSec 2008, Cambridge, MA, USA, September 15, 2008, Proceedings (LNCS 5210)*, pages 1–17, 2008.

[6] Fabian Fischer, Florian Mansmann, Daniel A. Keim, Stephan Pietzko, and Marcel Waldvogel. Large-Scale Network Monitoring for Visual Analysis of Attacks. In John R. Goodall, Gregory Conti, and Kwan-Liu Ma, editors, *Visualization for Computer Security – 5th International Workshop, VizSec 2008, Cambridge, MA, USA, September 15, 2008, Proceedings (LNCS 5210)*, pages 111–118, 2008.

[7] Ben Fry. *Visualizing Data*. O'Reilly, 2007.

[8] André Ricardo Abed Grégio. Aplicação de Técnicas de *Data Mining* para a Análise de *Logs* de Tráfego TCP/IP. Dissertação de Mestrado em Computação Aplicada do *Instituto Nacional de Pesquisas Espaciais*, 2007. Publicada e disponível na biblioteca do INPE.

[9] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[30] `http://www.octave.org/`
[31] `http://www.scilab.org/`
[32] `http://www.r-project.org/`
[33] `http://www.cs.waikato.ac.nz/ml/weka/`
[34] `http://rapid-i.com/`

[10] Alfred Inselberg. *Parallel Coordinates – Visual Multidimensional Geometry and Its Applications* . Springer, 2009.

[11] Md. Enamul Karim, Andrew Walenstein, Arun Lakhotia, and Laxmi Parida. Malware phylogeny generation using permutations of code. *Journal of Computer Virology*, 1(1):13–23, 2005.

[12] Daniel Keim. Visual Data Mining. Tutorial, *23rd International Conference on Very Large Data Bases (VLDB '97)*, 1997. Visitado em Agosto de 2009.

[13] Daniel A. Keim and Hans-Peter Kriegel. Using Visualization to Support Data Mining of Large Existing Databases. In John P. Lee and Georges G. Grinstein, editors, *Database Issues for Data Visualization – IEEE Visualization '93 Workshop, San Jose, California, USA, October 26, 1993, Proceedings (LNCS 0871)*, pages 1–17, 1994.

[14] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 2nd edition, 1997.

[15] Raffael Marty. *Applied Security Visualization*. Addison Wesley, 2008.

[16] Iosif-Viorel Onut, Bin Zhu, and Ali A. Ghorbani. SVision: A Network Host-Centered Anomaly Visualization Technique. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *Information Security – 8th International Conference, ISC 2005, Singapore, September 20-23, 2005, Proceedings (LNCS 3650)*, pages 16–28, 2005.

[17] Niels Provos and Thorsten Holz. *Virtual Honeypots*. Addison Wesley, 2008.

[18] Daniel Shiffman. *Learning Processing – A Beginner's Guide to Programming Images, Animation, and Interaction*. Morgan Kaufmann, 2008.

[19] Ben Shneiderman. Tree Visualization with Tree-maps: A 2-D Space-Filling Approach. *ACM Transactions on Graphics*, 11:92–99, 1991.

[20] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphic Press, 2nd edition, 2001.

[21] Xiang-Hui Wang and Guo-Yin Zhang. Web-Based Three-Dimension E-Mail Traffic Visualization. In Heng Tao Shen, Jinbao Li, Minglu Li, Jun Ni, and Wei Wang, editors, *Advanced Web and Network Technologies, and Applications – APWeb 2006 International Workshops: XRA, IWSN, MEGA, and ICSE, Harbin, China, January 16-18, 2006, Proceedings (LNCS 3842)*, pages 979–986, 2006.