UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

VISUALIZANDO ALERTAS PARA APRIMORAR O MONITORAMENTO DE SEGURANÇA EM REDE

TRABALHO DE GRADUAÇÃO

Diogo João Cardoso

Santa Maria, RS, Brasil

2015

VISUALIZANDO ALERTAS PARA APRIMORAR O MONITORAMENTO DE SEGURANÇA EM REDE

Diogo João Cardoso

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS) como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Raul Ceretta Nunes

409
Santa Maria, RS, Brasil

2015

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

VISUALIZANDO ALERTAS PARA APRIMORAR O MONITORAMENTO DE SEGURANÇA EM REDE

elaborado por
Diogo João Cardoso

como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA

Raul Ceretta Nunes, Dr.
(Presidente/Orientador)

Roseclea Duarte Medina, Profa. Dra. (UFSM)

Carlos Raniery Paula dos Santos, Prof. Dr. (UFSM)

Santa Maria, 11 de dezembro de 2015.

RESUMO

Trabalho de Graduação

Curso de Ciência da Computação

Universidade Federal de Santa Maria

VISUALIZANDO ALERTAS PARA APRIMORAR O MONITORAMENTO DE SEGURANÇA EM REDE

AUTOR: DIOGO JOÃO CARDOSO

ORIENTADOR: RAUL CERETTA NUNES

Data e Local de Defesa: Santa Maria, 11 de dezembro de 2015.

Este trabalho apresenta a proposta de uma ferramenta para visualização de alertas, que utiliza um banco de alertas de segurança de rede e ferramentas visuais voltadas a demonstração de dados. O banco de alertas pode ser gerado por qualquer ferramenta ou Sistema de Detecção de Intrusão (IDS), porém deve exportar os alertas no formato IDMEF. Neste trabalho o IDS Snort foi o escolhido para alimentar esse banco, pois é uma ferramenta bem flexível na configuração e suporta o formato de saída definido.

Com a ajuda de ferramentas de visualização de dados a ferramenta implementada procura alcançar uma consciência situacional da rede e a visualização dos eventos de segurança. As ferramentas visuais escolhidas foram Google Visualization API e Highcharts API, por serem implementadas em JavaScript e disponibilizarem uma vasta documentação de suas funcionalidades.

A visualização habilita uma percepção do contexto atual da rede pelo analista e auxilia na tomada de decisão. A ferramenta resultado permite que o analista visualize os alertas de intrusão com a possibilidade de conciliar outras informações sobre a rede, podendo melhorar a ação resposta à intrusão.

Palavras-chave: Monitoramento de segurança em rede, IDMEF, Visualização

ABSTRACT

Undergraduate Final Work

Undergraduate Program in Computer Science

Federal University of Santa Maria

VISUALIZING ALERTS TO IMPROVE NETWORK SECURITY MONITORING

AUTHOR: DIOGO JOÃO CARDOSO

ADVISOR: RAUL CERETTA NUNES

Defense Place and Date: Santa Maria, December 11th, 2015.

This work presents a proposal of a tool to visualize alerts, which uses a network security alerts database and visual tools that focus on visualizing data. The database can be generated by any Intrusion Detection System (IDS), but it has to export the alerts in IDMEF. For this work the chosen IDS to generate the database was Snort, because it has some configuration flexibility and supports the chosen output format.

With the help of data visualizing tools, the dashboard seeks to reach a situational awareness of the network and the ability to visualize security events. The visual tools chosen were Google Visualization API and Highcharts API, both are implemented in JavaScript and provide a vast documentation of its features.

The visualization enables a perception of the network current context by the analyst, and assists in the decision making. The resulting dashboard allows the analyst to see intrusion alerts with the possibility of reconcile other network info, enhancing the reply action to the intrusion.

Key-words: Network security monitoring, IDMEF, Visualizing

SUMÁRIO

1. INTRODUÇÃO

Com a popularização de serviços e aplicações web, o tráfego de dados na rede tem aumentado significativamente, e com isso o número de informações e dados críticos utilizados em rede. Em conjunto com o número de dados, o número de ataques que exploram vulnerabilidades dessas aplicações também aumentou, exigindo buscas e desenvolvimentos de ferramentas para combater essas invasões.

Sistemas de Detecção de Intrusão (*Intrusion Detection System*, IDS) monitoram a rede ou sistema de arquivos em busca de atividades maliciosas ou violações de política, coletando dados de alerta sobre essas atividades suspeitas e gerando relatórios para o administrador da rede (KOZZIOL, 2003). Uma das saídas de um IDS é a identificação de uma suposta invasão e, para isso, pode-se usar um padrão de representação de alertas como o IDMEF (*Intrusion Detection Message Exchange Format*).

O desenvolvimento desse formato padrão permite a interoperabilidade entre sistemas comerciais, *open source* ou de pesquisa.

Uma das principais ferramentas utilizadas para monitoramento da infraestrutura de redes e identificação de ataques são os chamados Sistemas de Consciência Situacional (KIZZA, 2005). Tais sistemas integram diferentes ferramentas com o objetivo de fornecer a um operador uma melhor consciência situacional sobre a situação do seu sistema alvo (ONWUBIKO e OWENS, 2011).

Neste contexto, nota-se que para complementar efetivamente um IDS é necessária uma ferramenta que mostre os alertas de forma visual. Um mecanismo com esse objetivo tem bastante utilidade quando uma rede gera muitos alertas, facilitando o monitoramento e possível tomada de decisão sobre algum tipo de alerta gerado, o que melhora a consciência situacional sobre a segurança da rede.

A consciência situacional aplicada a uma rede de computadores corresponde ao conhecimento do que está acontecendo em seu fluxo. Informações em conjunto podem auxiliar a obtenção dessa consciência, que deve ser utilizada pela equipe de monitoramento de segurança em rede na tomada de decisões. O processo para obter essa consciência situacional é geralmente dividido em três etapas: percepção, compreensão e projeção. A ideia por trás dessas etapas pode ser aplicada em um IDS, ou outra ferramenta que gere alertas de intrusão, com o intuito de obter a

consciência situacional da rede focando na segurança. Entretanto, salienta-se que a construção da consciência situacional é um processo de construção de conhecimento que extrapola em muito a identificação de correlações de eventos (KRUEGEL et al., 2004), requerendo o apoio de ferramentas visuais.

Logo, para complementar essa segurança, a visualização se torna um fator importante, podendo ser chamada de visualização de eventos de segurança. Ferramentas com este foco acabam trazendo algumas vantagens como: conhecimento sobre comportamentos e propriedades emergentes que não foram antecipados; melhor entendimento das características de grande e pequena escala; formação facilitada de hipóteses, entre outros.

Na tentativa de obter resultados melhores no monitoramento de segurança em rede, pode-se usar vários sistemas que irão coletar os eventos a serem monitorados. Uma boa implementação na visualização desses eventos de segurança ajuda na correlação entre os sistemas coletores e a equipe de monitoramento.

Neste trabalho é abordado esse aprimoramento visual no monitoramento de segurança em rede. Tendo como princípio a interpretação de um banco de dados composto de alertas no formato IDMEF, e a exploração de ferramentas visuais para uma consciência situacional, visa-se habilitar e uma melhor visualização de eventos de segurança.

O texto está organizado da seguinte forma. O capítulo 2 apresenta uma revisão bibliográfica abordando conceitos importantes para a realização desse trabalho. O capítulo 3 apresenta a estrutura proposta como solução neste trabalho. O capítulo 4 apresenta as implementações, resultados e análises feitas. O capítulo 5 apresenta as conclusões do trabalho.

1.1 Objetivo Geral e Específicos

O objetivo geral é explorar a área de monitoramento de segurança em redes, focando na representação de alertas visuais de uma forma eficiente para representar "meta-alertas". Com a especificação e implementação uma ferramenta que interpreta dados no formato IDMEF e realiza visualização com ferramentas de prateleira.

Objetivos específicos compreendem:

- Explorar o IDMEF e IDS;

- Utilizar ferramentas para visualização em conjunto com Consciência Situacional de rede;
- Implementar uma ferramenta para auxílio a monitoramento de segurança de redes.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo serão abordados conceitos importantes para a compreensão deste trabalho.

2.1 SISTEMA DE DETECÇÃO DE INTRUSÃO

Um Sistema de Detecção de Intrusão (*Intrusion Detection System*, *IDS*) é qualquer hardware, software ou combinação dos dois que monitora um sistema ou rede em busca de atividades maliciosas. Hoje em dia é um componente importante na arquitetura de segurança de redes. Seguindo a estratégia de um sistema de alertas comum, o IDS é colocado em locais comuns de entrada ou saída. Focando no que se dá como pontos fracos da estrutura do sistema, os quais podem ser considerados os pontos mais vulneráveis à ataques de invasão. (KOZZIOL, 2003)

Um IDS é constituído em três componentes funcionais: fonte de informação, análise e resposta. Um simples coletor de dados pode constituir a fonte de informação, e é responsável por capturar informações do tráfego da rede para serem utilizados pelo analisador. Essa análise, outro componente do IDS, é feita para encontrar sinais que indiquem a tentativa ou a ocorrência de uma intrusão, utilizando mecanismos de comparação. Por último, o componente de resposta é o responsável por executar as medidas de reação às intrusões com base nos resultados da análise, essas reações podem ser contra-ataques, bloqueio de recursos ou um simples alarme ao administrador da rede (NORTHCUTT; NOVAK, 2002).

Hoje em dia existem dois tipos de IDSs: os baseados em Rede, e os baseados em Host. Os IDSs baseados em Host residem em uma máquina e monitoram essa máquina em específico para tentativas de invasão (KIZZA, 2005). Já os IDSs baseados em Rede, que são mais populares, monitoram o tráfego de várias máquinas da rede. Um tipo não é necessariamente melhor que o outro, cada um é mais apropriado para situações específicas.

Os IDSs baseados em Host (*Host-Based IDSs*, *HIDSs*) monitoram ataques à nível de sistema operacional, aplicação ou kernel. Os HIDSs têm acesso à *audit logs* (logs de eventos), mensagens de erros, aos direitos dos serviços e aplicações, aos recursos disponíveis naquela máquina sendo monitorada, entre outros. Um HIDS

também pode trabalhar no nível de aplicação, tendo acesso aos seus dados e sabendo identificar quando um dado é anormal. Podendo analisar esses dados enquanto a aplicação executa (NORTHCUTT; NOVAK, 2002).

Os HIDSs são bons para determinar se um ataque foi bem sucedido ou não. O tráfego malicioso pode ser bem parecido com o tráfego comum, portanto é bem comum IDSs baseados em Rede gerarem alertas falsos. Por outro lado, o HIDS é bem mais preciso, pelo motivo de não gerar tantos falsos positivos quantos os IDSs baseados em Rede.

Já os IDSs baseados em Rede (*Network-Based IDSs, NIDSs*) são colocados em pontos chaves da infraestrutura da rede, para monitorar o tráfego em tempo real entre os nós. A utilização de IDSs baseados em Rede cresceu bastante, passando a taxa de uso e aceitação de HIDSs. Um NIDS tem um custo benefício melhor que um HIDS, pois cobre a segurança de vários aparelhos dentro da infraestrutura da rede, diferente da utilização um pra um do HIDS. Assim utilizando NIDS um analista da rede tem uma ampla visão sobre o que está acontecendo na rede. E o monitoramento de algo mais específico, um host ou atacante, pode ser aumentado ou diminuído conforme necessário (KOZZIOL, 2003).

Existem várias maneiras em que o IDS detecta invasões na rede, algumas técnicas são boas em detectar vários tipos de invasões, e outras técnicas trabalham em tipos mais específicos. Um IDS normalmente implementa várias técnicas para ser mais abrangente na detecção. As técnicas mais comuns são Detecção por Assinatura, Detecção por Anomalia e Verificação de Integridade.

Na Detecção por Assinatura, são identificados eventos que tentam executar de uma maneira não convencional. Representações de invasões conhecidas desses eventos são chamadas de regras e são guardados no IDS. As regras são comparadas com as atividades futuras do sistema. Assim que alguma atividade corresponda a uma invasão conhecida é gerado um alerta ao analista do sistema (KOZZIOL, 2003).

Na Detecção por Anomalia, o sistema trabalha com a ideia de que um ataque tem um comportamento diferente do definido pelo sistema (KRUEGEL; VALEUR; VIGNA, 2004). É analisado um período de tempo e comparado com um padrão de comportamento. Se alguma atividade acontece fora desse padrão é então gerado um alerta. Esse tipo de detecção é comum no monitoramento das atividades de usuários, e é bem útil na detecção de ataques mais sofisticados (KOZZIOL, 2003).

Já a Verificação de Integridade, trabalha utilizando checksums para determinar se houve mudança nos arquivos do sistema ou não. Comparando com o checksum dos arquivos originais, sempre que o valor der diferente ele gera um alerta. Essa técnica é bastante usada em serviços Web, pois mostra se um atacante fez alguma mudança no conteúdo das páginas. Um detalhe importante é que deve-se ter cuidado para não gerar falsos positivos, portanto o checksum deve ser sempre corretamente atualizado (KOZZIOL, 2003).

2.2 SNORT

Criado em 1998 por Marty Roesch, o Snort tinha como objetivo ser uma ferramenta de auxílio a análise de tráfico de rede (KOZIOL, 2003). Eventualmente sendo liberado ao público, Snort evoluiu suas funcionalidades se tornando uma ferramenta muito prática na área de segurança de rede. Podendo ser utilizado com *sniffer*, *packet logger* ou IDS baseado em Rede.

Um sniffer pode ser definido como uma aplicação que coleta todos os dados que passam na rede. O Snort pode executar em modo sniffer, capturando os dados destinados a outros hosts dentro da rede. Isso é possível devido à utilização de canal de dados compartilhados da arquitetura Ethernet, que tem como objetivo diminuir o custo do tráfego. O Snort executando em moto sniffer mostra todos os dados coletados na tela.

Já em modo packet logger, o Short coleta os dados de maneira parecida a um sniffer, porém os dados são gravados em logs. Esses logs podem ser gravados utilizando texto ASCII ou tcpdump, sendo o tcpdump o mais recomendado para redes de tráfego alto e a performance é um fator importante.

O Snort também opera em modo NIDS, que assim como o modo sniffer e packet logger, coletando todos os dados que passam na rede. A diferença é o que fazer com os dados coletados, em modo NIDS o Snort analisa cada pacote e determina se é benigno ou maligno. Gerando alertas quando encontra algum tráfego suspeito.

O Snort possibilita com a utilização de plug-ins a geração de logs de saída dos alertas detectados. Esses logs normalmente são gerados com o intuito de serem analisados pela equipe de monitoramento de segurança de rede. Porém nem sempre tem um formato humanamente legível, e são utilizadas ferramentas externas

para sua interpretação. Assim, o Snort se torna bem flexível em relação aos dados gerados, deixando a escolha e formatação a cargo da equipe que utilizará a ferramenta.

Os logs de saída podem ser gerados nos seguintes formatos:

- syslog
- tcpdump
- Log de Texto
- XML
- Banco de Dados Relacional
- SNMP
- Short Unified

Considerando essa capacidade dos IDSs de gerar logs em formatos pré-definidos, é possível a criação de um banco de alertas de intrusão. Onde esse banco é alimentado por um ou vários IDSs e pode ser utilizado para análise dos eventos de segurança da rede. Assim, existe uma correlação entre as ferramentas para a criação de um banco de alertas unificado e uma abstração de quem cria o alerta, já que todos seguem o mesmo padrão.

2.3 CONSCIÊNCIA SITUACIONAL

A Consciência Situacional (SA, *Situational Awareness*) envolve o conhecimento sobre o que está acontecendo na sua vizinhança. É também uma área de estudo voltada à percepção do ambiente, o que é crítico para tomadores de decisão de áreas complexas da aviação, controle de tráfego aéreo, usinas de energia, comandos e controle militar, e também serviços de TI (Tecnologia e Informação) (ONWUBIKO; OWENS, 2011).

A definição tradicional descreve predições relacionadas ao tempo, espaço, referências e objetos. Um exemplo seria o piloto de uma aeronave, que tem que levar em conta não só o estado de seu próprio veículo (velocidade, altura, latitude e longitude), mas também a localização de aeronaves próximas, locais de pouso entre outros. Quando se fala de Consciência Situacional na área de Tecnologia da Informação, entra a definição de *Cyber Situational Awareness* (Cyber-SA) (ONWUBIKO; OWENS, 2011). Tendo o mesmo intuito de obter o conhecimento

sobre seu ambiente, Cyber-SA leva em consideração a topologia da rede e suas características, o que pode se tornar complexo de manejar. Portanto existe a divisão em três estágios (ONWUBIKO, OWENS, 2011):

- Percepção: inclui a detecção e reconhecimento de dados que levam a consciência de situacional da rede;
- Compreensão: ou a consciência de comportamento malicioso e como impactarão o ambiente atual da rede;
- Projeção: avaliação da possibilidade de futuras ações resultadas da situação atual da rede;

2.4 O FORMATO IDMEF

Criado pela IDWG (*Intrusion Detection Working Group*), o IDMEF (*Intrusion Detection Message Exchange Format*) (DEBAR; CURRY; FEINSTEIN, 2007) é um formato de dados padrão que sistemas de detecção de intrusão utilizam para reportar e compartilhar alertas sobre eventos considerados suspeitos (alertas). O desenvolvimento desse formato padrão permite a interoperabilidade entre sistemas comerciais, open source ou de pesquisa a integrarem os seus sistemas de acordo com seus pontos fortes e fracos para obterem uma otimização da implementação (DEBAR et al., 2007).

A utilização mais comum de IDMEF é entre um IDS (*Intrusion Detection System*) e um controlador ou console para o qual o IDS manda os alarmes. Outros exemplos onde IDMEF pode ser útil:

- Um sistema com banco de dados único que recebe e guarda resultados de vários IDSs proveria a possibilidade de analisar os dados e tomar decisões em cima de toda a situação, não somente parte dela;
- Um sistema de correlação de eventos que recebe alertas de diversos IDSs seria capaz de realizar confirmações de cálculos e correlações cruzadas (*cross-correlations and cross-confirmations calculations*) mais sofisticadas do que um sistema limitado a um IDS;
- Uma interface gráfica para um usuário, que exibe alertas providos de várias IDSs, possibilita ao usuário monitorá-los em uma só tela. E o usuário só teria que aprender uma interface, ao invés de várias.

- Um padrão no formato de dados facilitaria não somente a troca de dados entre organizações, mas também a sua comunicação.

Duas implementações do IDMEF foram originalmente propostas à IDWG, uma utilizando SMI (*Structure of Management Information*) para descrever uma Base de informações de gerenciamento (MIB) utilizando o protocolo SNMP (*Simple Network Management Protocol*). E a outra utilizando DTD para validar documentos XML. E em fevereiro de 2000, foi decidido que a implementação utilizando XML seria a utilizada, pois era a que mais cumpria os requisitos da IDWG.

A saída no formato IDMEF possui uma classe base para todo o modelo (*IDMEF-Message*). A classe *IDMEF-Message* por sua vez deriva em duas classes especializadas, *Alert* e *Heartbeat,* que agregam uma série de classes (DEBAR et al., 2007). O relacionamento entre as classes é apresentado na Figura 2.1.

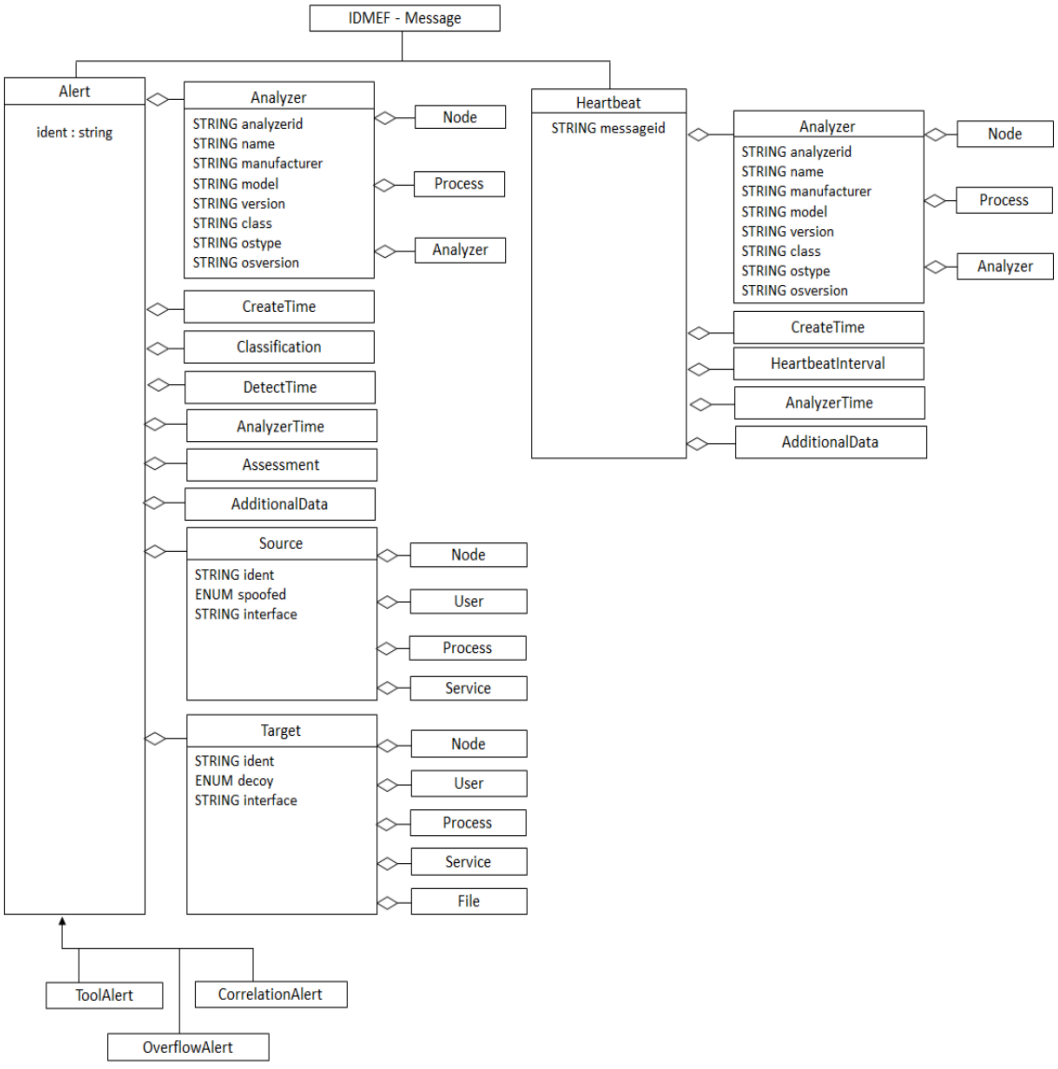

Figura 2.1: Visão geral do formato IDMEF (adaptado de (SÁ BRANDÃO, 2007)).

Para melhor entendimento e utilização do formato precisa-se descrever cada classe. Com o conhecimento de quais informações o IDMEF disponibiliza, é possível analisar os alertas e retirar informações do que está acontecendo na rede. Abaixo segue a descrição dos elementos principais do alerta, segundo (IDMEF RFC):

- <Alert>: dependendo do sensor, uma mensagem *Alert* pode corresponder a um ou vários eventos detectados. Os alertas ocorrem de forma assíncrona em resposta a eventos externos;
- Analyser: informações para identificação do analisador que gerou o alerta;
- CreateTime: Momento em que o *Alert* ou *Heartbeat* foi criado. Dos três dados de tempo que podem aparecer em um alerta, este é o único obrigatório;
- DetectTime: Momento em que o evento que produziu o alerta foi detectado. No caso de mais de um evento terá o valor do primeiro detectado, assim nem sempre tem o mesmo valor que o CreateTime. O analisador não precisa gerar o alerta no momento em que o evento foi detectado;
- AnalyserTime: O momento de envio da mensagem de alerta pelo analisador;
- Source: A fonte do evento(ou eventos) que gerou o alerta;
- Target: O alvo do evento(ou eventos) que gerou o alerta;
- Classification: o 'nome' do alerta;
- Assessment: Informações no impacto do evento que gerou o alerta, ações tomadas pelo analisador em resposta ele, e o nível de confiança na avaliação das informações fornecidas pelo analisador;
- AdditionalData: Informações inclusas pelo analisador que não encaixam com o formato original do IDMEF.
- <Heartbeat>: Representa o estado atual do analisador. *Heartbeats* são normalmente programados para serem enviados em um intervalo predefinido. E o recebimento de um *Heartbeat* significa que o analisador está funcionando corretamente.

2.5 GOOGLE VISUALIZATION API

Uma das ferramentas grátis disponibilizadas pelo Google é o *Google Visualization API*, que junto com o *Google Chart API* são ferramentas visadas na geração de gráficos. O *Google Chart API* trabalha mais com gráficos estáticos, já o *Google Visualization API* trabalha com gráficos interativos, sendo de fácil integração a um painel ou site interativo. Uma da principais características dessas ferramentas é a flexibilidade não só nos gráficos gerados, mas na fonte dos dados utilizados. Os gráficos podem ser personalizados de diversas formas (cores, formatos e disposição dos objetos), e a fonte de dados podem ser valores gerados na hora, lidos de um arquivo ou a partir de alguma outra ferramenta externa (*Google Spreadsheet*, *Google Fusion Tables*, entre outras).

Mesmo sendo bem personalizável, a API disponibiliza uma galeria de gráficos base que contém os tipos mais frequentemente utilizados como: gráfico de pizza, linhas, colunas, *treemap*, área, dentre outros. Tendo também uma documentação bem completa, já que possui um leque de funcionalidades bem grande.

O funcionamento da ferramenta se dá através da utilização do *Google AJAX API*, que inclui várias outras APIs do Google. Para ter acesso ao *Google Visualization API* é necessário utilizar o método *google*.*load*(), recebendo como parâmetro a versão desejada. Uma chamada de função call-back executa assim que a API é carregada, essa função contém as informações e definições de configuração desejadas. O código abaixo demonstra essa chamada:

```
google.load('visualization','1',{'packages':['corechart']}
);
google.setOnLoadCallback(drawChart);
```

A função *google*.*load*() vai carregar a *API Visualization* na versão 1.0, e chama o pacote ‘*corechart*’ que é o padrão a ser utilizado. Esse pacote define qual chart (gráfico) será desenhado, utilizando o padrão *corechart* pode-ser ter acesso a todos os tipos de *charts*. A função 'drawChart' é a função de call-back que irá fazer todas as definições do gráfico.

A partir do pacote padrão, é definido a estrutura de dados a ser utilizada. Para isso é definida uma variável que irá utilizar a função google.visualization.DataTable(). Nessa variável são adicionadas as linhas e colunas do suposto gráfico. Nessa hora pode-se também definir outras configurações mais específicas, como cores, largura

e altura. Os tipos de dados que pode-se trabalhar pode variar com o gráfico escolhido, a grande maioria utiliza sem problemas valores numéricos, preferencialmente em ordem. Discretizar valores é uma tática bem comum para melhorar a distribuição dos dados, e até mesmo ajudar em alguma maneira de utilizar *strings* como valores nos eixos.

Por fim o *chart* é carregado utilizando uma função específica dependendo qual o tipo desejado, o código abaixo representa a chamada de um gráfico do tipo Pizza:

```
var chart = new google.visualization.PieChart(..);
chart.draw(data, options);
```

A função *PieChart* recebe como argumento o objeto algo no HTML que irá receber o gráfico. Os dados definidos na variável *data* e *options* são passados como parâmetro também para a geração do gráfico.

2.6 HIGHCHARTS

A ferramenta *Highcharts* é uma API criada pela *Highsoft AS*, empresa localizada na Noruega, e é famosa por ser utilizada por sites famosos como *Facebook*, *Twitter* e *Yahoo*!. A empresa criou várias ferramentas em *JavaScript* com o propósito de complementar aplicações web. Outros exemplos de ferramentas criadas pela *Highsoft AS* são: *Highstock JS* que se trata de uma API focada em gráficos para visualização financeira; *Highmaps JS* é uma ferramenta para utilização de mapas interativos para mostrar vendas ou resultados de eleições por exemplo; e o próprio *Highcharts JS* que é uma ferramenta para geração de gráficos interativos em *JavaScript*. O *Highcharts* disponibiliza uma biblioteca de *charts* comuns e são bem personalizáveis. A empresa também disponibiliza uma documentação completa para suas ferramentas.

O *Highcharts* utiliza AJAX para executar portanto é necessário importar as bibliotecas necessárias. A ferramenta trata seus dados como conjuntos (chamados de *series*) e cada conjunto pode ter características específicas (título, legenda, eixos, cores entre outros). Os diferentes gráficos disponibilizados pela *Highsoft* têm configurações e aspectos diferentes no tratamento de dados. As configurações mais simples vão de cores e fonte, à *tooltip* personalizada e comandos que executam ações extras.

2.7 VISUALIZAÇÃO

A visualização de dados pode ser definida como uma maneira moderna de comunicação visual. Ela não faz parte de uma área específica, pois se encaixa em várias, como estatística descritiva ou criação de teorias através de análises. E envolve a criação e estudo de representações de visualização de dados, definidos como informações que podem ser abstraídas de forma esquemática, incluindo atributos ou variáveis das unidades de informação (FRIENDLY, 2008).

Este trabalho foca na visualização de dados de tráfego de rede. Segundo Sheffler (2002), a ideia por trás da visualização de dados em análise de tráfego é que os dados devem ser mostrados ao usuário num formato que permita fácil compreensão, e facilite a identificação de tráfego e padrões anômalos.

Segundo Mark Green (1998), é mais fácil para o ser humano perceber informações dispostas em uma maneira espacial do que não espacial. Partindo desse conceito é necessário explorar a visualização espacial, como por exemplo os eixos XY e até mesmo o Z.

Porém imagens com mais de três variáveis, duas espaciais e uma não espacial, não são tão eficazes para se identificar padrões (BERTIN, 1983). Portanto é necessário utilizar maneiras de certa forma simples na visualização, este trabalho procura explorar essas características fazendo combinações de cores e planos na visualização dos dados.

3. ESTRUTURA DA SOLUÇÃO

Tendo como foco o trabalho com um banco de alertas que utiliza um formato predefinido, e a exploração as ferramentas visuais com o intuito de visualizar os alertas de rede, este capítulo mostra as ferramentas utilizadas (seção 3.1 e 3.2) e a estrutura da solução proposta (seção 3.3).

3.1 BANCO DE ALERTAS

Como visto anteriormente o Snort é uma ferramenta flexível e condiz com as necessidades propostas nesse trabalho. Na implementação foi definido que o Snort executará em modo IDS, para gerar alertas de intrusão, e será a ferramenta que irá alimentar o Banco de Alertas a ser utilizado.

Uma das características que levou a escolha do Snort é o suporte a vários formatos de saída que podem ser utilizados em seus logs. O formato IDMEF é característico na utilização de logs de alertas de intrusão, e foi o formato alvo na implementação. Portanto foi utilizado um *plugin* para o Snort que permite a geração de logs no formato desejado.

As versões mais recentes do Snort (2.9.7.6 sendo a última na execução deste trabalho) não suportam o *plugin* do IDMEF pois foi descontinuado alguns anos atrás. Segundo os desenvolvedores da ferramenta, o motivo da escolha foi que haviam poucos usuários utilizando o formato.

Na implementação deste trabalho foi utilizada a versão 2.8.3.2 do Snort, lançada em 2009. A ferramenta foi instalada em um Linux Ubuntu 12.04 e algumas bibliotecas para o funcionamento foram necessárias como libpcap, uma bibioteca para captura de tráfego de rede implementada em C/C++; e bibliotecas auxiliares para a utilização do *plugin* IDMEF do Short: libidmef e libxml2.

3.1.1 ESTRUTURA DO BANDO DE ALERTAS

O princípio da utilização de um Banco de Alertas é abstrair do usuário quem está detectando as intrusões e gerando os alertas. A ideia inicial é que uma ou mais

ferramentas externas possam alimentar esse banco para fins de portabilidade e variedade de detecções, pois cada IDS pode trabalhar de uma maneira diferente.

A figura 3.1 mostra um exemplo de saída IDMEF gerado pelo Snort.

```
<IDMEF-Message version="1.0">
   <Alert ident="289">
   <Analyzer analyzerid="109" model="snort" version="2.8.3.2">
      <Node>
      <name>tcpdump_dmz</name>
      </Node>
   </Analyzer>
   <CreateTime ntpstamp="0xc36cc187.0xd3aa9b49">2007-11-
24T17:42:31Z</CreateTime>
   <Source>
      <Node>
         <Address category="ipv4-addr">
            <address>135.013.216.191</address>
         </Address>
      </Node>
      <Service>
         <port>22</port>
         <protocol>tcp</protocol>
      </Service>
   </Source>
   <Target>
      <Node>
         <Address category="ipv4-addr">
            <address>172.016.112.149</address>
         </Address>
      </Node>
      <Service>
         <port>22</port>
         <protocol>tcp</protocol>
      </Service>
   </Target>
   <Classification origin="vendor-specific">
      <name>msg=(spp_stream4) STEALTH ACTIVITY (NULL scan) detection</name>
      <url>none</url>
   </Classification>
   </Alert>
</IDMEF-Message>
```

Figura 3.1: Exemplo de IDMEF exportado pelo Snort.

Como visto na seção 2.4 os campos mais relevantes na utilização do IDMEF são: <Alert>, Analyser, CreateTime, DetectTime, AnalyserTime, Source, Target, Classification, Assessment, AdditionalData e <Heartbeat>. Com base nesse conhecimento, foi necessário estudar e avaliar quais desses dados, combinados, formam uma boa maneira de visualizar os alertas.

3.2 VISUALIZAÇÃO

Com a criação do Banco de Alertas e a definição de quais informações retiradas dos eventos serão utilizadas, é necessário estudar paradigmas de visualização que trabalham com alertas de rede.

A tarefa de monitorar os riscos de segurança e atividades maliciosas em uma rede é uma tarefa importante e há ferramentas que auxiliam esse monitoramento. Porém ferramentas como varredores de vulnerabilidades, *firewalls* e IDSs geram logs de saída em forma textual e pacotes de captura de tráfego de rede (pcap). Esses arquivos tendem a crescer, e com arquivos de saída vindo de múltiplas fontes rapidamente torna-se difícil a tarefa de analisar os alertas e tomar ações de resposta de maneira eficiente. Para superar esse desafio, pesquisadores têm investigado maneiras de converter representações textuais de *datasets* de rede em representações visuais em duas dimensões (2D), com o intuito de melhorar o monitoramento de rede a partir de *datasets* maiores (KEIN, 2002).

A visualização 2D se baseia na utilização dos eixos X e Y, que são usados para identificar, detectar e analisar informações maliciosas . Porém, a medida que o número de alertas em uma rede for crescendo, mostrar as informações de maneira clara e visível se torna uma tarefa cada vez mais complicada (ARK; DRYER; SELKER, 1998). Portanto, para complementar a visualização 2D, existe também a visualização em três dimensões (3D). Adicionar o eixo Z na avaliação, permite um aprofundamento melhor na informação sendo visualizada e os resultados podem ser percebidos de uma maneira mais clara.

Para este trabalho algumas técnicas de visualização foram exploradas usando as ferramentas *Google Visualization* e *Highcharts*. As definições e exemplos de utilização de algumas técnicas exploradas serão apresentados a seguir.

3.2.1 TÉCNICAS DE VISUALIZAÇÃO 2D

A visualização 2D se baseia na utilização de um contexto entre o eixo X e Y, onde usualmente um desses eixos representa o tempo.

3.2.1.1 SCATTER PLOTS

A técnica de *Scatter Plots* utiliza pontos cujos os dois valores X e Y representam dois atributos. Essa técnica pode ser utilizada para identificar padrões na relação desses atributos. Um exemplo de trabalho é o NVisionIP que utiliza *scatter plots* para auxiliar analistas da rede a visualizar o seu estado atual. A figura 3.2 representa uma rede de classe B e suas subnets, onde cada ponto representa um IP sendo monitorado que contém algumas informações úteis ao analista (LAKKARAJU; YURCIK; LEE, 2004).

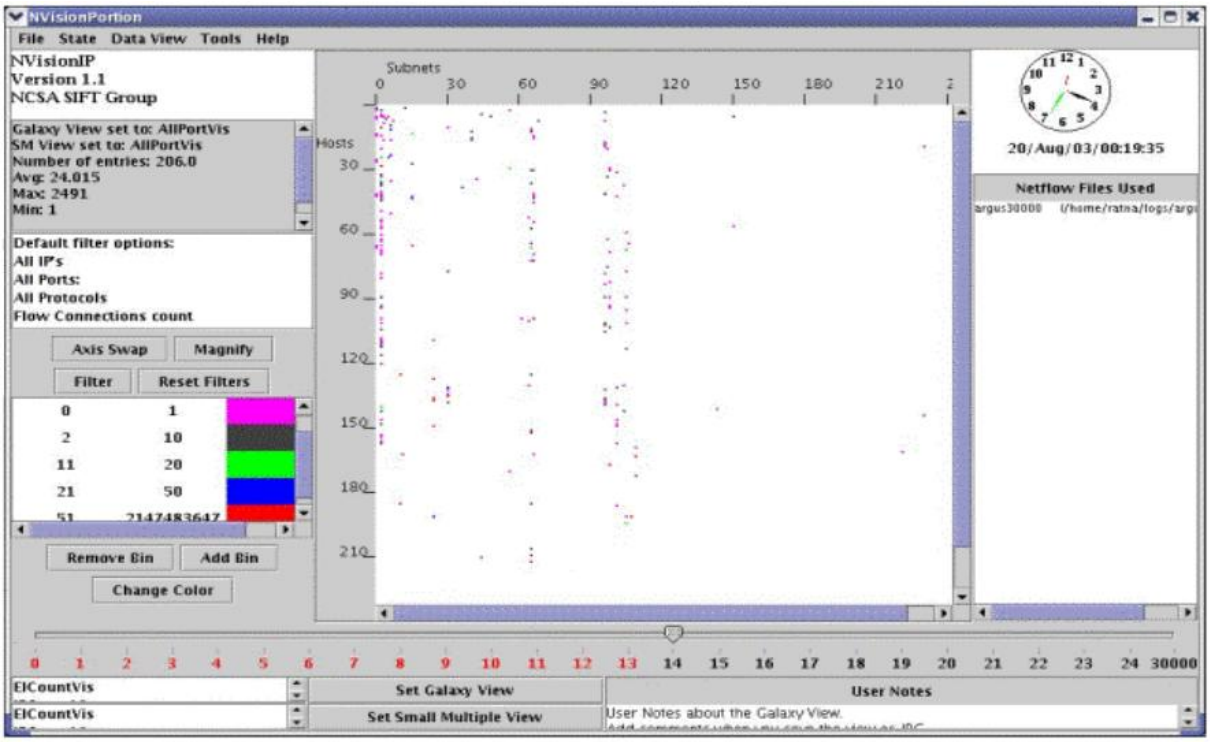

Figura 3.2: NVisionIP Galaxy View (Adaptado de (LAKKARAJU et al., 2004)).

3.2.2 VISUALIZAÇÃO 3D

A visualização 3D utiliza, além dos eixos X e Y, também o eixo Z para adicionar mais informação. Nesse caso cada um dos eixos pode representar um parâmetro do dado. Assim, o número de dados sendo representados e a diversidade de combinações de atributos são grandes. Um exemplo de aplicação que utiliza visualização 3D em rede é o Spinning *Cube of Potential Doom* (LAU, 2004). A ferramenta utiliza *scatter plots* com a terceira dimensão, representando o IP destino,

a porta destino e o IP fonte nos eixos X, Y e Z respectivamente. Os pontos são representados em cores diferentes, onde cada cor representa um tipo de conexão (UDP ou TCP por exemplo). A partir dessa configuração é possível identificar alguns comportamentos na rede, a figura 3.3 demonstra o *Cube of Doom* em um caso de *Port Scan*.

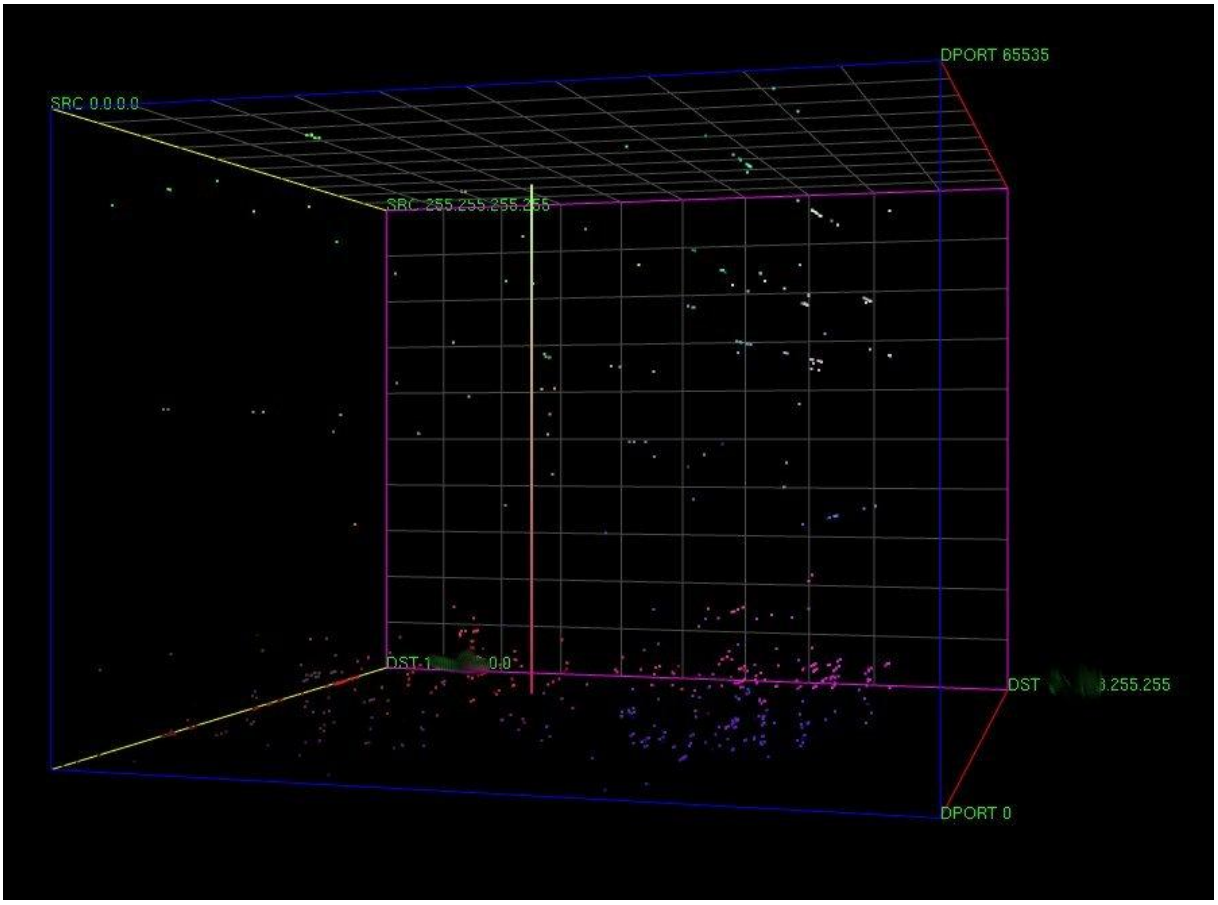

Figura 3.3: Demonstração de um Port Scan no Cube of Doom (YIN et al., 2007).

3.3 ESTRUTURA DA SOLUÇÃO PROPOSTA

O problema alvo é a visualização de alertas de rede utilizando as ferramentas visuais Google API e Highcharts tendo como foco suas potencialidades de visualização e as possibilidades de interação que o analista de rede pode ter. Com o Banco de Alertas definido (seção 3.1) e o conhecimento de algumas de técnicas visualização de alertas (seção 3.2) é possível estruturar a solução para o problema alvo.

A solução contém um Painel de Controle (*Dashboard*) que faz a leitura do Banco de Alertas, no formato XML, e mostra as informações obtidas. Uma das

características buscadas na implementação foi que a ferramenta possa gerar uma espécie de contexto em cima das informações obtidas, ou seja, permitindo um aprofundamento na visualização ou uma filtragem nos dados por exemplo. Esse contexto tem o intuito de aprimorar o monitoramento atual da rede.

O funcionamento da ferramenta foi proposto da seguinte maneira: inicia-se lendo o arquivo correspondente ao banco de alertas e a partir dele gera-se algumas listas das informações básicas que serão utilizadas. Essas listas podem conter todos os IPs fonte ou IPs alvo por exemplo. Assim a informação pode ser guardada de forma compacta e de fácil acesso.

Utilizando as listas geradas os vários gráficos devem processá-las e mostrar os diferentes resultados. É bem importante para questão de desempenho, que a ação de ler o arquivo (e gerar as listas) seja separada da ação de ler as listas (e gerar os gráficos). Pois são operações caras e não necessariamente devem acontecer com a mesma frequência.

A partir disso os gráficos processam separadamente as informações desejadas, gerando a visualização. Cada tipo de gráfico pode ter uma preferência na informação que irá utilizar, um exemplo disso é poder determinar a cor pelo tipo de protocolo (TCP, UDP, etc). Pode ser que algum gráfico nem utilize a informação do protocolo.

Uma das características da ferramenta é permitir que o usuário tenha uma certa profundidade na utilização, permitindo a visualização de dados mais específicos, ou informações extras à que está visível no gráfico. Outra característica é que os gráficos possam ser correlacionados. A correlação serve para que a visualização seja consistente, para que todos os gráficos tenham um mesmo objetivo. Assim as informações de vários gráficos podem ser utilizadas em conjunto.

Por fim vale ressaltar que a consciência situacional que a ferramenta propõem ocorre a partir da utilização das opções extras aos gráficos e suas interações com o analista. Utilizando essas informações extras é possível analisar um novo caso de maneira melhor. Um exemplo simples seria a possibilidade de guardar um histórico de um *host*. Assim quando se for analisar alertas em um período de tempo é possível ver se esse comportamento já aconteceu antes, podendo influenciar na ação tomada.

4. PROJETO E IMPLEMENTAÇÃO

Este capítulo apresenta o detalhamento da proposta de implementação de um *Dashboard* para visualização de alertas de rede, utilizando as ferramentas *Google Visualization* e *Highcharts*. A seção 4.1 apresenta a estrutura de dados que será explorada na visualização e suas características. A seção 4.2 e 4.3 apresentam as funcionalidades do *Google API* e *Highcharts* respectivamente, suas limitações e os resultados obtidos. A seção 4.4 apresenta a proposta e implementação da correlação dos gráficos e consciência situacional buscada neste trabalho.

A metodologia utilizada na implementação foi a seguinte, primeiro se estudou as ferramentas de visualização e quais técnicas seriam possíveis de se aplicar. Com isso foi necessário montar as estruturas de dados que seriam utilizadas e testar com exemplos de alertas. Na implementação do *dashboard* foi necessário interligar os dados e configurações dos gráficos e criar variáveis e contextos que seriam utilizados.

4. 1 ESTRUTURA DE DADOS

A partir da leitura do arquivo XML foi necessário definir uma estrutura de dados com as informações que serão exploradas. Na implementação foram utilizadas uma série de listas, com o intuito de impedir a releitura do banco de alertas inteiro para realizar buscas.

Como visto anteriormente um alerta carrega algumas informações a serem utilizadas, são elas: o tempo de criação (*timestamp*), IP e porta da fonte (*source*), IP e porta do alvo (*target*), tipo do protocolo e a classificação da gravidade do alerta (por exemplo, *low*, *medium* ou *high*). A partir desse conhecimento as seguintes listas foram criadas:

- Lista 1: contém todos os *IPs source*, sem elementos repetidos;
- Lista 2: contém todos os *IPs target*, sem elementos repetidos;
- Lista 3: contém todos os alertas gerados, com as informações citadas anteriormente;

Nota-se que na Lista 3, caso algum alerta contenha mais de um *source* ou *target*, gera-se várias linhas na lista com cada combinação, mantendo sempre uma relação 1 para 1.

A partir dessas listas, é dever da ferramenta de visualização interpretar os dados e gerar os gráficos desejados. As seções a seguir demonstram como as ferramentas trataram a estrutura de dados e geraram seus resultados.

4. 2 GOOGLE API

Como visto no capítulo 2, a ferramenta do Google utiliza duas variáveis para gerar cada gráfico. A variável *data* contém todos os dados a serem mostrados, normalmente em forma de tabela e com uma ordem específica que define os eixos e características do gráfico. E a variável *options*, que contém as configurações de visualização como, cores, títulos, formato de texto, etc.

Por limitações da própria ferramenta, o Google API não suporta a utilização do eixo Z. Portanto os exemplos explorados na implementação com esta API utilizam somente duas dimensões.

4.2.1 BUBBLE CHART

O primeiro gráfico estudado foi o *Bubble Chart*, que se caracteriza por relacionar os eixos X e Y no plano cartesiano. O diferencial é que o *bubble chart* utiliza alguns dados extras para diferenciar o tamanho de cada *bubble* (obrigatório) e até mesmo a cor (opcional).

Na implementação do *Bubble Chart* as informações utilizadas foram os IPs *source* e *target*, e o número de alertas para cada combinação (*source* x *target*). Usando as listas 1 e 2 (definidas em 4.1) faz-se uma varredura em todos os alertas do banco e uma contagem de quantos alertas correspondem a cada combinação. Uma matriz resultado é gerada com todos os valores iniciais ‘null’ e para uma lista de N alertas, serão feitas N incrementações nessa matriz. O resultado final é definido com uma DataTable() e guardado na variável *data* que é passada como parâmetro para a função draw().

A estrutura da matriz resultado fica a seguinte:

```
[ [“IP Fonte”, “IP Alvo”, “Alertas”]
 [0,  0,    9],
 [0,  1,    5]
… ]
```

A primeira coluna representa os IPs *source* e a segunda coluna representa os IPs *target*, sendo que os valores na tabela correspondem ao respectivo índice na lista de IPs. O motivo de se usar números que representam índices é que o *Bubble Chart* não suporta *strings* como valores dos eixos, somente números. Isso acaba dificultando um pouco a utilização pois limita que tipo de informação pode ser definida como eixos.

A figura 4.1 demonstra um exemplo de resultado obtido pelo *Bubble Chart*, onde cada valor nos eixos representa um IP e as bolhas representam a existência de alertas, sendo o tamanho definido pela quantidade que foi lida.

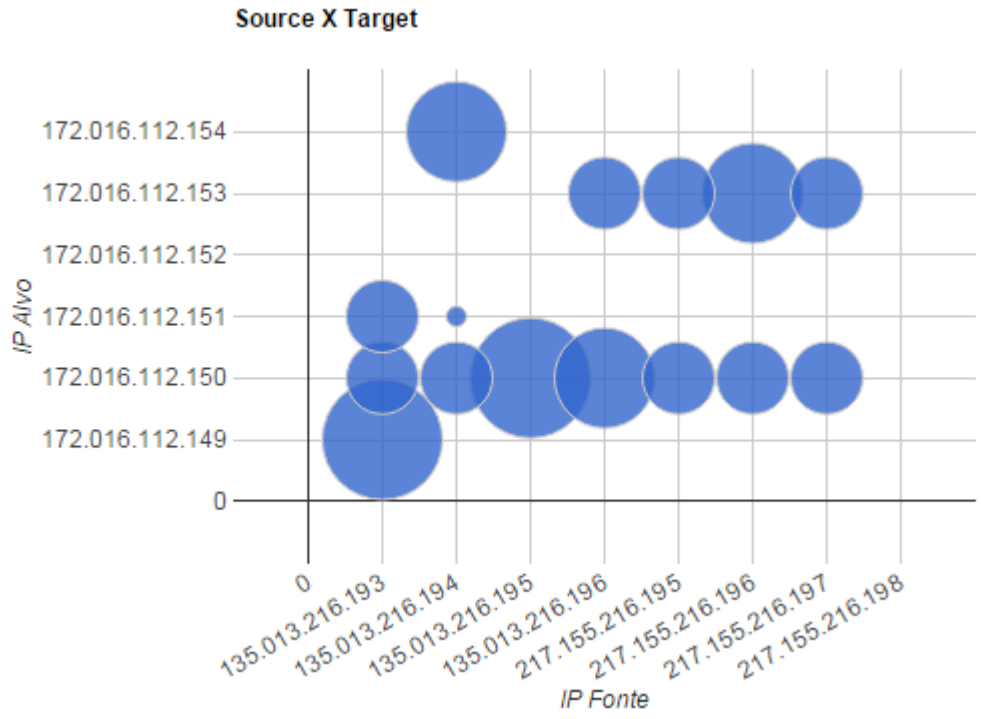

Figura 4.1: Exemplo de visualização de alertas com o *Bubble Chart*.

Uma ideia para complementar essa implementação pode ser a utilização da cor da bolha para representar uma informação extra. Por exemplo, mostrar a média da gravidade dos alertas que cada bolha representa (muitos alertas e todos com classificação '*Low*' ainda forma uma bolha grande, porém a gravidade não é alta). Outro exemplo da utilização da cor pode ser os diferentes tipos de protocolo,

identificando se o comportamento dos alertas em uma máquina alvo é sempre o mesmo, mostrando se várias máquinas geraram os mesmos tipos de alerta num mesmo alvo.

Por fim, uma das desvantagens desse gráfico é que é o um dos únicos da API do Google que não suporta o uso de *tooltip* personalizável, um ferramenta que permite informações extras quando o usuário passa o mouse em uma bolha específica, limitando um pouco a interação que poderia ser feita com o usuário.

4.2.2 SCATTER CHART

O *Scatter Chart* que o Google API implementa também trabalha com somente 2 dimensões. A definição dele é bem parecida com o *Bubble Chart*, porém ele trabalha com pontos e eles não variam de tamanho. Uma das vantagens do *Scatter Chart* sobre o *Bubble Chart* é que não exige um terceiro parâmetro além do X e Y na visualização, deixando-o mais simples de se usar.

Também foi utilizado um DataTable() para montar os dados a serem visualizados. Mas diferente do *Bubble Sort*, não existe uma coluna que define a cor, e sim cada coluna representa o que é chamado de 'série'. Cada série é representada por uma cor, e cada linha da tabela preenche a coluna alvo com um dado e as restantes com 'null'.

O resultado é uma matriz com três colunas apenas, onde a primeira coluna representa a hora do alerta e a segunda e terceira definem se o protocolo é TCP ou UDP, respectivamente. O valor da coluna que representa o protocolo correspondente é o número da porta destino, sendo a coluna do outro protocolo 'null'.

A seguir um exemplo da matriz resultado:

```
[ [new Date(2015,10,24,9,59),22,null],
  [new Date(2015,10,24,10,00),22,null],
  [new Date(2015,10,24,10,01),null,22],
... ]
```

A princípio é mostrado todas as portas destino de todos os *targets.* Posteriormente será explorada a filtragem desses dados.

A figura 4.2 mostra as portas destino dos alertas em certo período de tempo. Pode-se identificar o protocolo de cada alerta pela cor gerada. A partir de um gráfico

como esse pode-se identificar padrões e os comportamentos nos destinos dos alertas. Talvez uma porta sendo constantemente alvo seja alguma vulnerabilidade; ou se vários *sources* geram alertas em sequências de portas pode significar tentativas de *port scan*. Ou até mesmo identificar que alguma porta específica não gera muitos alertas, podendo ter algum significado para o contexto da rede

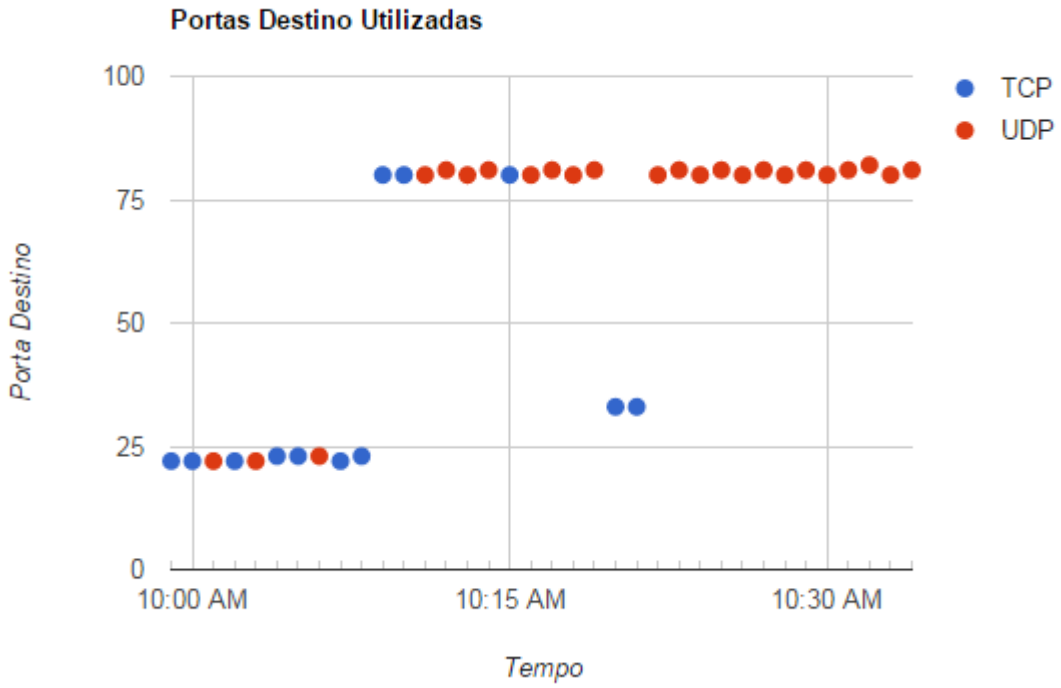

Figura 4.2: Exemplo de visualização da utilização de portas com o *Scatter Chart*.

Como complemento, o *Scatter Chart* permite o uso de *tooltip* personalizável, assim pode-se adicionar qualquer tipo de informação para ser mostrada quando o usuário passa o mouse nos pontos.

4.2.3 SANKEY DIAGRAM

O *Sankey Diagram*, diferente dos citados anteriormente, não utiliza o plano 2D e sim um fluxo entre os dados definidos. Cada entidade sendo conectada é chamada de node, e cada conexão é chamada de link. A conexão entre dois nodes pode ter um peso, salientando a visualização sobre os outros links. A ferramenta automaticamente organiza os nodes e links e adiciona níveis a mais, caso necessário.

Para este caso para montar a tabela de dados é necessário somente os dois campos que serão ligados, e um peso atribuído a essa ligação. Na implementação foi definido que cada nó representa um IP e a conexão representa o número de alertas gerados entre os dois. O DataTable() gerado tem o formato parecido com o utilizado no *Bubble Chart*, duas colunas representam os IPs e a terceira representa o número de alertas. Porém o *Sankey Diagram* consegue trabalhar com *strings*, então optou-se por utilizar o IP inteiro ao invés de um índice para uma lista.

A seguir um exemplo da matriz resultado:

```
[ ["135.013.216.193","172.016.112.149",4],
  ["135.013.216.193","172.016.112.150",2],
  ["135.013.216.194","172.016.112.154",3]
... ]
```

A figura 4.3 mostra o Sankey Diagram aplicado na visualização da relação *source-target* dos alertas. Note que o Sankey Diagram tem uma certa semelhança com o *Bubble Chart* no exemplo implementado, pois os dois demonstram a relação *souce-target*. Porém por não depender de um plano 2D, o *Sankey Diagram* pode prover uma percepção diferente. É possível analisar de forma mais eficiente de onde vem o maior fluxo de alertas sendo gerado, e qual é o alvo mais comum. E em um exemplo de 3 ou mais níveis, é possível identificar relações mais isoladas.

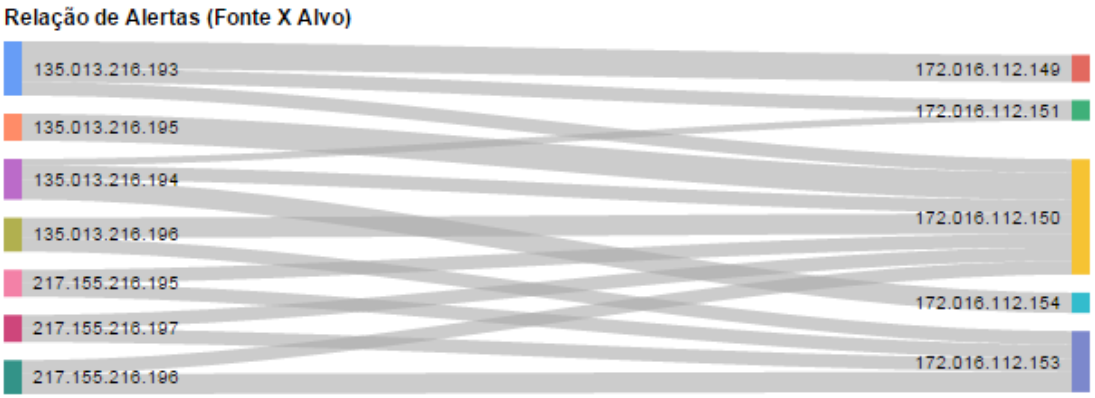

Figura 4.3: Exemplo de visualização com o Sankey Diagram.

4.3 HIGHCHARTS

Como visto no capítulo 2, o *Highcharts* utiliza conjuntos (chamado de *series*) para definir seus dados. Cada gráfico exibe de maneira diferente esses conjuntos,

mas a estrutura de todos é similar. Na definição o gráfico em si utiliza um vetor de series, e cada serie tem alguns atributos (name, data, tooltip, color, size, label, etc) .

4.3.1 BAR CHART

O primeiro exemplo implementado em *Highcharts* foi um gráfico de barras simples, mas com o objetivo de ser flexível à interação com o usuário. O dado alvo na implementação foi a quantidade de alertas que cada *source* conhecido gerou nos hosts da rede (*target*). Para isso, no primeiro contexto do gráfico, o eixo X representa os hosts, e o eixo Y o número de alertas. Feita a contagem, o gráfico mostra uma barra em cada host, que representa um *source* e o número de alertas correspondente. Para a visualização geral, todas as barras de um mesmo *source* tem a mesma cor.

Na implementação do *highcharts*, os hosts são definidos pela opção 'category' nas configurações do eixo X e o resultado se torna uma lista de *series* com cores únicas. Abaixo um exemplo *serie*:

```
series: [{
      name: '135.013.216.194',
      data: [19,0,0,73,0,7],
      color: '# 871b47'
},
... ]
```

O código acima define um conjunto inteiro de barras, cada valor da variável *data* é o número de alertas para cada host. Os hosts são definidos em *categories*, e o índice dos dados é o que faz a ligação. Por exemplo, nesta *serie* o *source* gerou 19 alertas para o host 0, 73 alertas para o host 3, e assim por diante.

A figura 4.4 representa um possível resultado de visualização após a leitura de vários alertas. Nota-se a fácil identificação das cores e a concentração dos alertas em um dos hosts. O objetivo da implementação desse gráfico é servir de complemento às outras visualizações, pois tem uma interface simples e bem comum.

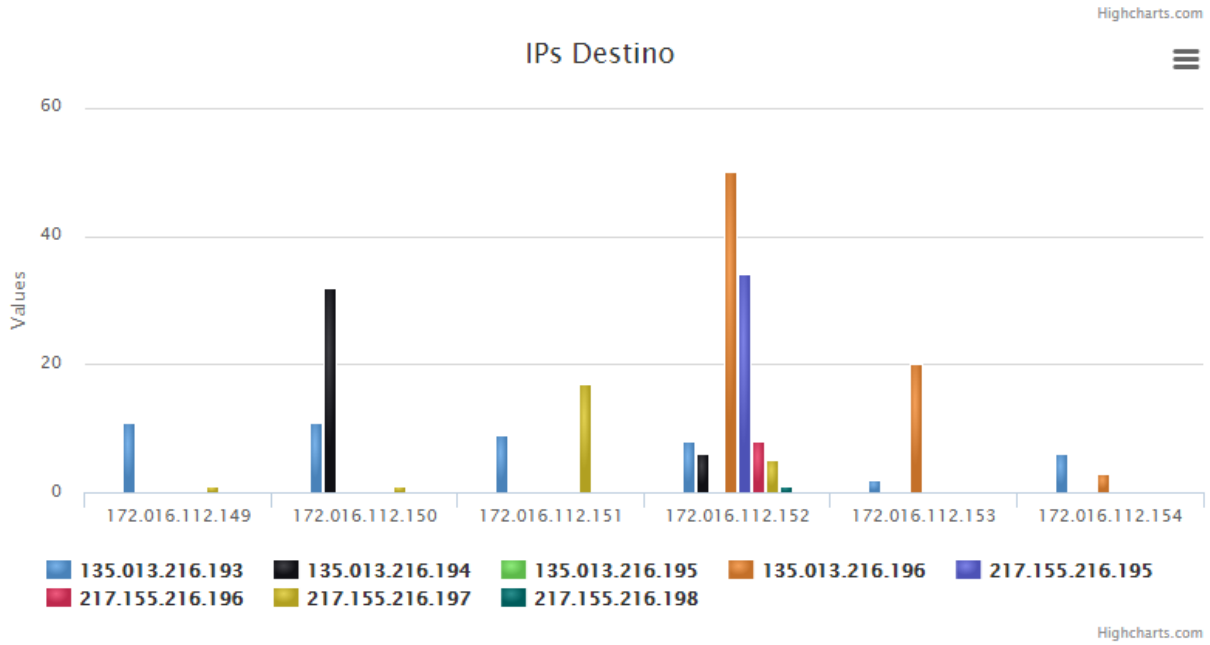

Figura 4.4: Exemplo de visualização de quantidade de alertas por host com o *Bar Chart*.

4.3.2 SCATTER CHART 3D

Diferente do *Google API*, o *Highcharts* tem suporte à utilização da terceira dimensão e é possível utilizar o eixo Z efetivamente. O *Scatter Chart 3D* é nada mais que o *Scatter* comum com a adição de um eixo a mais de dados. Para utilizar a terceira dimensão basta habilitar a opção 'options3d' nas configurações do gráfico.

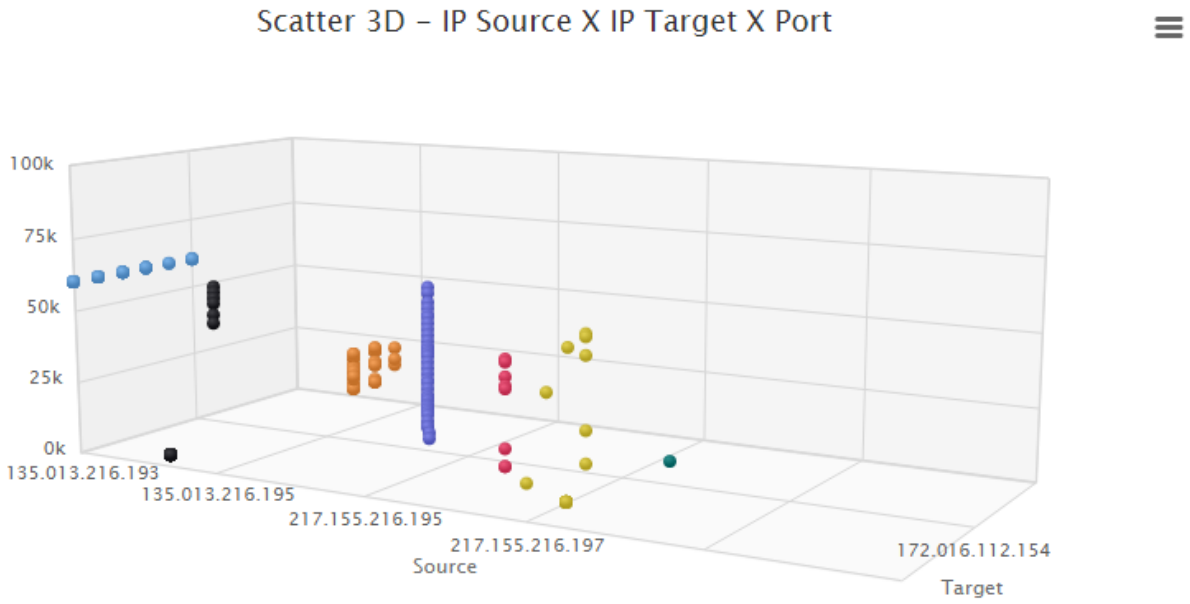

Figura 4.4: Exemplo de visualização da utilização de portas com o Scatter 3D.

O exemplo implementado é uma relação entre IP fonte, IP alvo e Porta utilizada em cada alerta. A partir das listas definidas em 4.1, gera-se uma matriz que alimentará o gráfico. E essa matriz contém 3 colunas que correspondem aos três atributos citados.

O que deve ser observado no exemplo são os padrões que os pontos formam. Linhas retas ou um aglomerado de pontos podem significar algo relevante. Por exemplo, na figura 4.4 o conjunto de pontos formando a linha roxa significa que o *source* gerou vários alertas a um *targert* específico, porém usando diferentes portas. O gráfico não demonstra essa informação mas normalmente esses alertas testam as portas em sequência, podendo significar um *port scan*.

Junto com o *Scatter Chart 2D*, a versão com 3 dimensões é uma das mais promissoras quando o objetivo é analisar comportamentos e padrões nos alertas. Essas duas são as mais utilizadas em ferramentas que utilizam visualização no gerenciamento de rede hoje em dia.

4.4 DASHBOARD

O objetivo deste trabalho não é somente visualizar alertas com as ferramentas vistas neste capítulo, mas também utilizá-las para alcançar uma possível consciência situacional. Deste modo, escolheu-se implementar um *Dashboard* que tem como objetivo unir o funcionamento de todos os gráficos, gerando uma coesão entre eles, e permitir que o usuário tenha uma certa profundidade na utilização das ferramentas.

A facilidade que o usuário vier a ter para identificar algum comportamento ou fluxo diferente é o que ditará a eficiência da ferramenta. A interação do usuário é importante e ter a habilidade de filtrar informações auxilia bastante na análise.

Esse foi o motivo da implementação das listas citadas em 4.1, com elas é possível criar uma consistência nos dados que todos os gráficos irão utilizar, e caso haja alguma mudança será aplicada em todos. Foi também implementado variáveis de controle que ditarão a filtragem dos dados quando necessário.

4.4.1 IMPLEMETAÇÃO

Uma das maiores vantagens que as ferramentas visuais escolhidas têm é que elas são implementadas em JavaScript. Portanto elas têm suporte à interações com o usuário e podem realizar ações personalizáveis. Nos exemplos implementados neste trabalho foram utilizados os chamados *listeners de eventos* que executam uma função após serem ativados. Para os gráficos implementados foi utilizado como disparador de eventos o *select*. Toda vez que o usuário seleciona uma representação de dado no gráfico (ponto, bolha, seção, etc) o *listener* é chamado e a função definida nele é executada. O *listener* tem acesso aos dados da seleção feita, portanto ele pode identificar o que foi selecionado e buscar mais informações se necessário.

Um exemplo de implementação do *Dashboard* é o mostrado na figura 4.5. Nela foram utilizados os gráficos *Scatter Chart* e *Scatter 3D*, em conjunto do *Sankey Diagram* e um Gráfico de Barras.

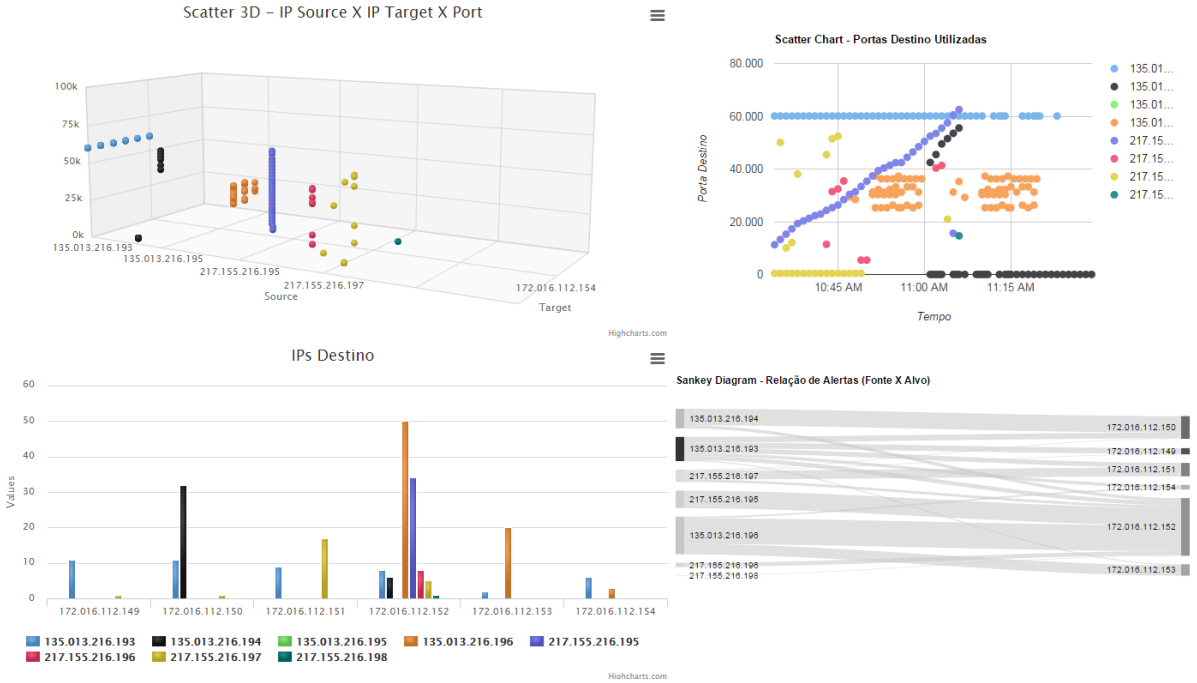

Figura 4.5: Exemplo de tela do *Dashboard* mostrando 4 diferentes gráficos.

Com esta visão o analista pode observar os dois gráficos *Scatter* a procura de padrões visuais nos pontos representados. As cores representadas nos dados são uma parte bem importante da visualização. Nesse exemplo os dados da mesma cor

representam o mesmo IP *Source*. Com exceção do *Sankey Diagram*, que não permite tal personalização nas cores.

Pode-se observar que nos gráficos *Scatter 2D* e *3D* é possível identificar padrões nos pontos. Vamos tomar como exemplo os pontos roxos, que representam o IP *Source* "217.155.216.195". No *Scatter 3D* observa-se que os pontos formam uma linha vertical, indicando que esse *source* gerou a maioria dos seus alertas para um único *target*. Observando os pontos roxos no *Scatter 2D*, nota-se que com o tempo a porta destino foi mudando, cada alerta gerado tem uma porta diferente. Pode-se usar o gráfico de barras para observar se realmente foi um só destino. Com essas informações o analista deve identificar que tipo de comportamento se trata.

Em uma ferramenta com propósito parecido, Alex Wood (2003) visualiza e identifica esse comportamento como sendo um *Fetchmail*. Que explora o *overflow* de *buffer* na versão 5.8.6 do linux.

Para se chegar a uma conclusão de qual ataque se trata os alertas sendo visualizados, é necessário conhecer as vulnerabilidades dos *hosts*, e que tipo de ataque é proeminente de acontecer.

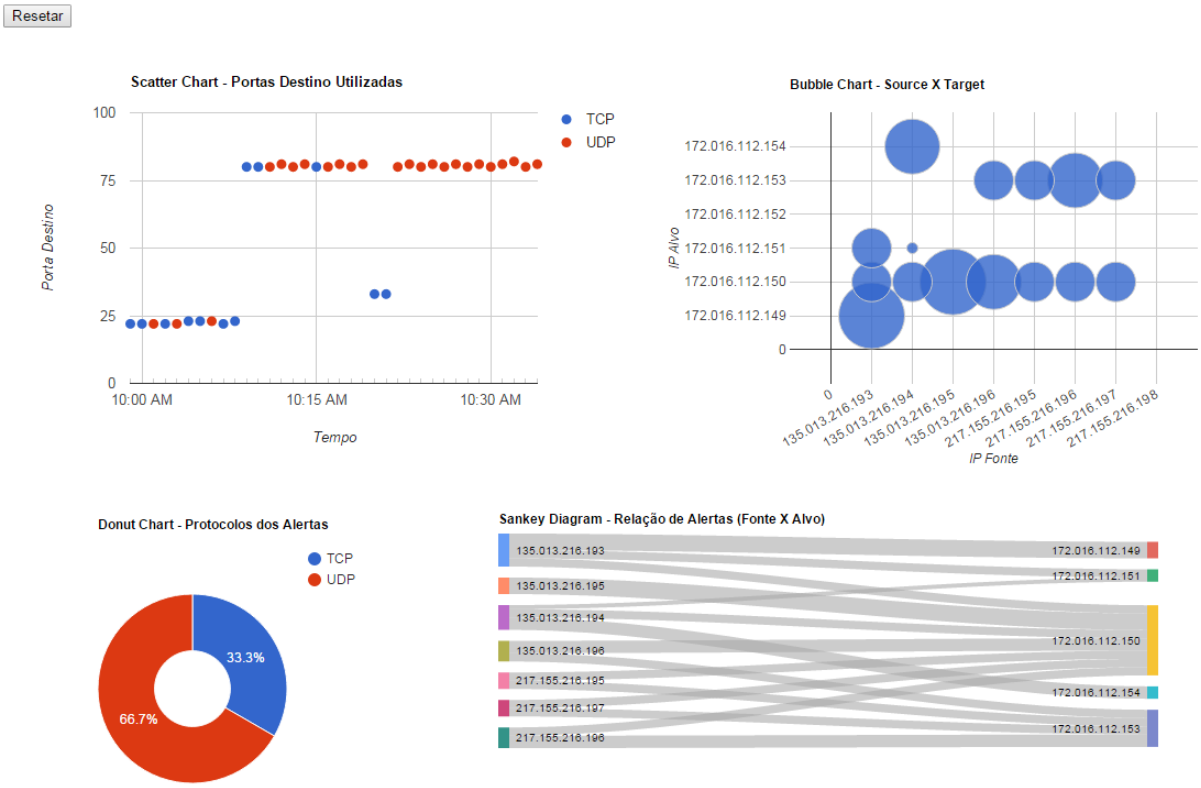

Figura 4.6: Novo exemplo de tela do Dashboard, habilitando interações com os gráficos.

A figura 4.6 demonstra outro exemplo do *Dashboard*. Nesse exemplo, foram implementadas algumas interações com os gráficos, que permitem uma visualização de dados mais específicos. Por exemplo, se o usuário clicar em algumas das barras coloridas do gráfico *Sankey Diagram*, todo o *Dashboard* irá filtrar para as informações relacionadas ao IP selecionado, podendo ser um IP *source* ou *target*. A figura 4.7 mostra o resultado ao clicar, na Figura4.6, no IP alvo "172.016.112.150".

Por questão de praticidade foi implementado também um botão para restaurar a configuração inicial do *Dashboard*, que fica no canto superior esquerdo (vide Figuras 4.6 e 4.7).

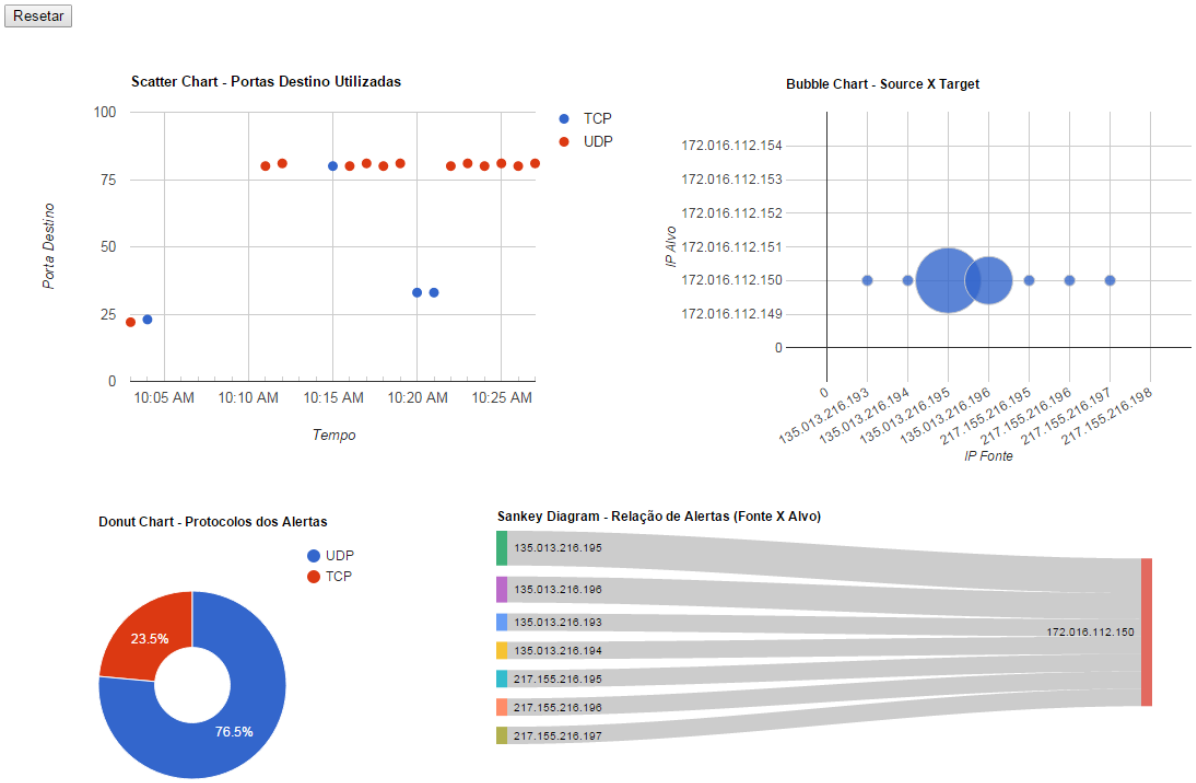

Figura 4.7: Visão geral do Dashboard com filtro de IP.

A figura 4.6 mostra outro exemplo que ilustra o tipo de interação que o usuário pode ter com o *Dashboard*. O gráfico da esquerda é o gráfico inicial, cada IP no eixo X representa um *host* da rede (*target*), e as barras representam os alertas gerados para cada um. Quando o usuário clica em um *host*, é mostrado um novo gráfico que expande essas informações. No exemplo ele simplesmente mostra de maneira mais clara cada um dos IPs *Source* que gerou alerta para o host em questão. Com a cor correspondendo aos mesmos IPs anteriormente mostrados, a figura 4.9 demonstra o resultado.

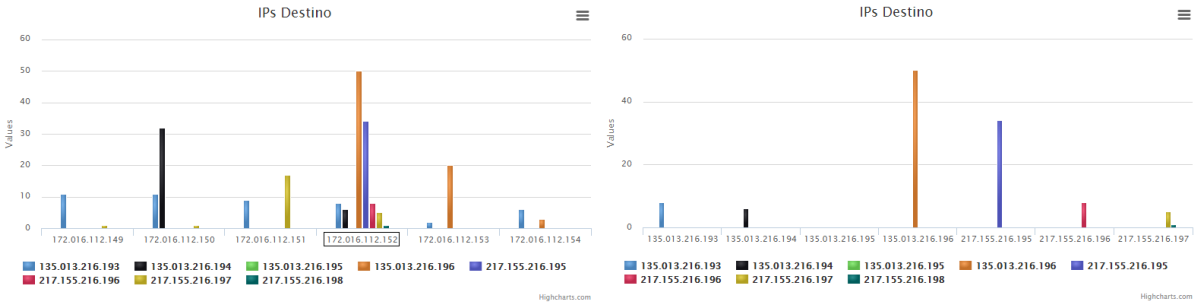

Figura 4.8: Interação do Gráfico de Barras.

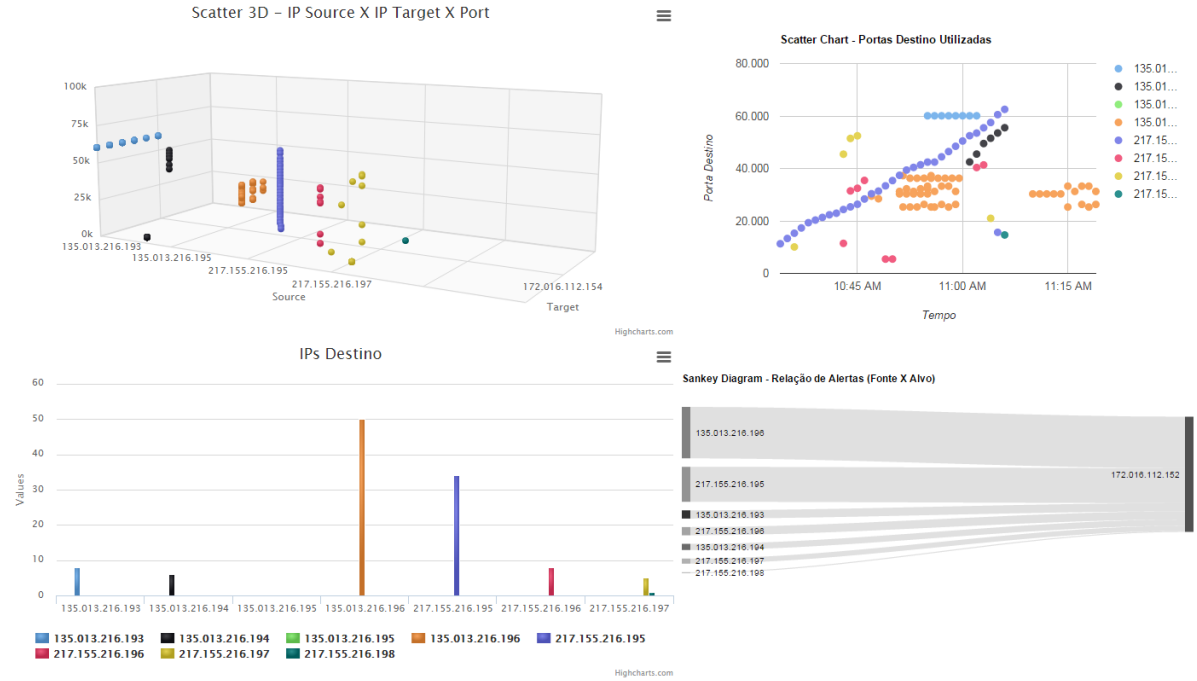

Figura 4.9: Visão do *Dashboard* com filtro de IP e interação com o gráfico.

Note que na figura 4.9 é mostrado uma visão do *Dashboard* com a filtragem do host selecionado (neste caso sendo o "172.016.112.152"). O *Sankey Diagram* e o *Scatter Chart* foram configurados para se ajustarem a esse filtro, mostrando somente informações relacionadas ao host selecionado. Pode-se observar as diferenças com a figura 4.5 vista anteriormente.

4.4.2 CONSCIÊNCIA SITUACIONAL

Para auxiliar o monitoramento da rede de forma eficiente pode-se buscar a implementação de uma consciência situacional. Um exemplo disso pode ser a classificação dos alertas de acordo com o impacto funcional, o impacto em informações ou até o quão difícil pode ser de se recuperar o sistema devido a alguma intrusão (CICHONSKI, 2012).

Esse tipo de classificação pode ser implementado na ferramenta por meio de cores, resultando em uma fácil captação das informações pelo analista.

A figura 4.10 utiliza o mesmo exemplo de dados visto anteriormente, porém classificando os alertas de acordo com a sua gravidade (ou impacto). O atributo utilizado é um componente da classe do alerta (sub-classe *classification*) que pode ser relacionado a um incidente na rede, necessitando ser avaliado.

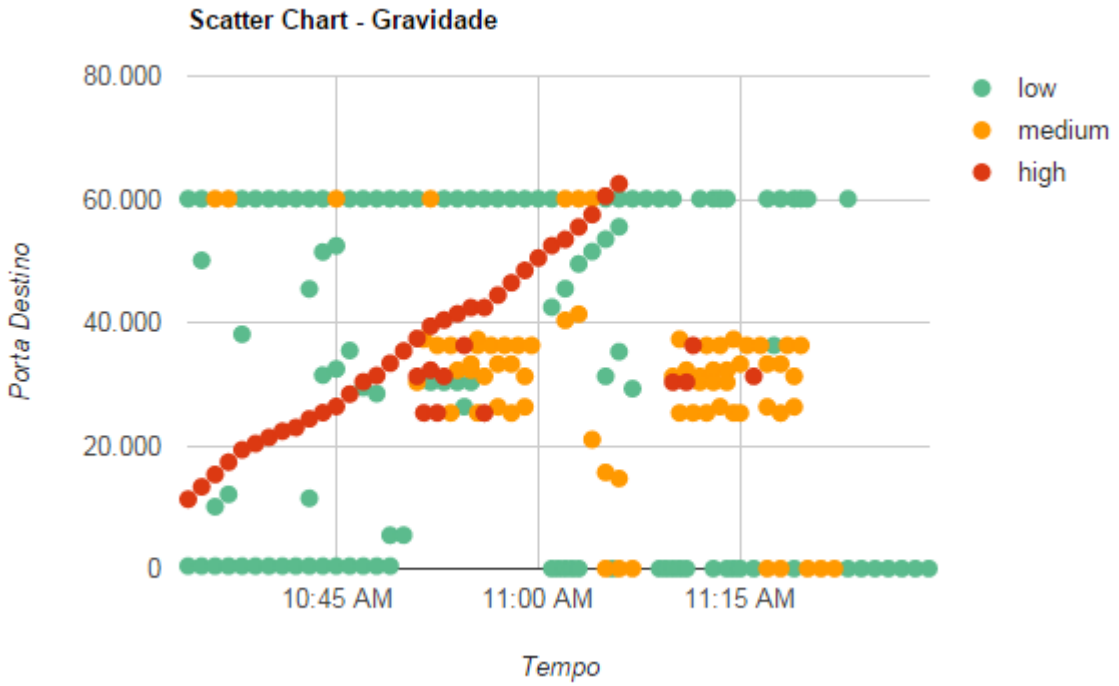

Figura 4.10: Visualização da gravidade dos alertas.

Segundo (CICHONSKI et. al., 2012), a gravidade dos alertas pode entrar em algumas categorias: o impacto funcional e o impacto de informação são exemplos comuns. O impacto funcional dita o quão impactante será a ação que gerou o alerta nos serviços envolvidos. As categorias para esse impacto são: baixa, média e alta.

Um alerta categorizado com impacto funcional baixo não tem um efeito muito grande, os serviços mais críticos podem ainda funcionar corretamente porém com menos eficiência. Já um categorizado com impacto alto irá comprometer seriamente os serviços providos e os usuários não poderão ser atendidos.

Para o *dashboard* implementado esse tipo de análise é um bom complemento às outras informações. A figura 4.11 demonstra a integração dessa análise à interação com o analista. Ao visualizar todos os alertas há a opção de ver os IPs *source* que os geraram ou a gravidade de cada um no determinado intervalo de tempo. Assim o analista pode saber se algum conjunto de alertas apresenta alguma ameaça pelo impacto que foram classificados.

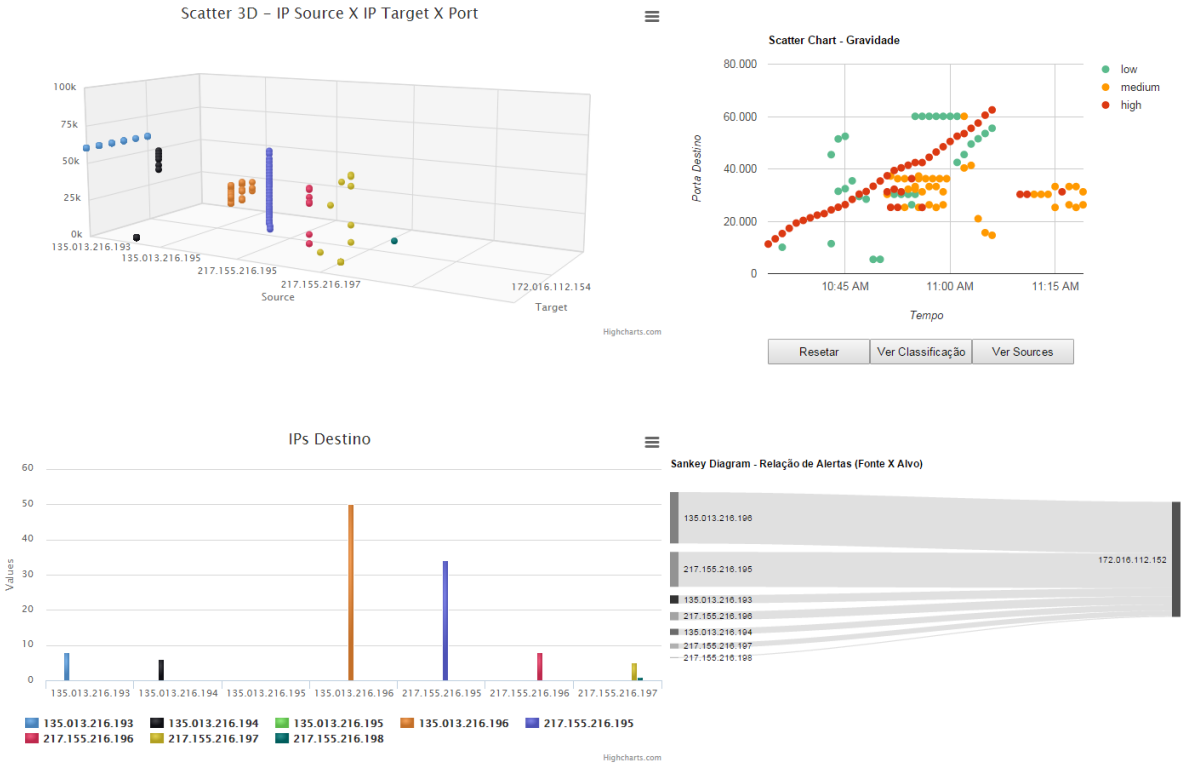

Figura 4.11: Exemplo de visão do Dashboard com a integração da gravidade dos alertas.

Na figura 4.11 o gráfico mostrando a classificação dos alertas substitui o gráfico anterior que mostrava os IPs *source*. Com a adição dos botões é possível fazer com que esses gráficos possam ser alternados a qualquer momento. Outros tipos de classificação podem ser adicionados, basta que a tag *classification* do banco de alertas tenha essa informação, o que depende do tipo alerta e do IDS que o gerou.

5. CONCLUSÕES

A visualização de alertas de rede pode se tornar importante na análise de ataques, e para que ocorra de maneira efetiva deve-se usar uma ferramenta que consiga facilitar e complementar esse objetivo. Porém uma ferramenta com essas características tende a ser paga e de código fechado, não permitindo uma personalização completa da parte do usuário.

Neste trabalho foram exploradas duas ferramentas gratuitas (*Google API* e *Highcharts*) com o intuito de visualizar alertas de rede. Como são ferramentas genéricas, as duas provaram que é possível utilizá-las, mas pelo mesmo motivo pode se tornar algo muito superficial.

Por isso pode-se concluir que para a utilização completa das potencialidades das ferramentas, é necessário que o programador intervenha e implemente as funcionalidades extras que virão a aperfeiçoar a visualização dos alertas de rede. E por isso pode-se dizer que as possibilidades de customização são inúmeras.

Neste trabalho pôde-se alcançar uma visualização de alertas que permitisse a identificação de padrões, porém essa identificação vem do analista e não da API em si. Isto exigindo uma implementação mais complexa de funções extras à API para se chegar a um resultado mais completo.

Para trabalhos futuros pode-se citar o desenvolvimento de uma ferramenta de visualização de alertas que explore de forma aprofundada a utilização de eventos do *Google API*. De forma que o usuário possa desenhar e interagir com a topologia da rede, obtendo uma visualização mais detalhada dos alertas gerados dentro da estrutura da rede.

REFERÊNCIAS

ABDULLAH, K.; LEE, C.; CONTI, G.; COPELAND, J. A.; STASKO, J.. **IDS RainStorm: Visualizing IDS Alarms**. Disponível em: <http://www.cc.gatech.edu/~stasko/papers/vizsec05.pdf>

ARK, W.; DRYER, C. D.; SELKER, T.; ZHAI S. **Representation Matters: The Effect of 3D Objects and a Spatial Metaphor in a Graphical User Interface.** Proceedings of HCI on People and Computers, pp. 209–219, 1998.

BERTIN, J. **The Semiology of Graphics.** Univ. Wisconsin Press: Madison, Wisc. 1983

CERT.BR. **Centro de Estudos, Resposta e Tratamento de Incidentes no Brasil.** Disponível em: <http://www.cert.br/>

CHOI, H.; LEE, H.; KIM, H. **Fast Detection and Visualization of Network Attacks on Parallel Coordinates**. Computers and Security, vol. 28, no. 5, pp. 276 – 288, 2009.

CICHONSKI, P.; MILLAR, T.; GRANCE, T.; SCARFONE, K. **Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology.** NIST Special Publication 800-61, 2012

CLAISE, B. **Cisco Systems NetFlow Services.** 2004.

DEBAR, H.; CURRY, D.; FEINSTEIN, B. **The Intrusion Detection Message Exchange Format (IDMEF)**. RFC 4765. March 2007

ERBACHER, R. **Intrusion behavior detection through visualization**. Systems, Man and Cybernetics. 2003. IEEE International Conference on, vol. 3, pp. 2507–2513 vol.3, 2003.

FRIENDLY, M. **Milestones in the history of thematic cartography, statistical graphics, and data visualization.** 2008

GOOGLE. **Charts – Google Developers.** 2015. Disponível em: <https://developers.google.com/chart/>.

GREEN, Marc PhD. **Toward a Perceptual Science of Multidimensional Data Visualization: Bertin and Beyond.**

GULA, R. **Correlating IDS Alerts with Vulnerability Information**. Tenable Network Security, Maio de 2011.

HIGHCHARTS. **Interactive JavaScript charts for your webpage – Highcharts.** 2015. Disponível em: <http://www.highcharts.com/>.

KEIM, D. **Information Visualization and Visual Data Mining** IEEE Transactions on Visualization and Computer Graphics, pp. 1–8, Mar 2002.

KIZZA, J. M. **A Guide to Computer Network Security**. New York, NY: Springer, 2005.

KOZIOL, J. **Intrusion Detection with Snort.** Sam Publishing 2003.

KRUEGEL, C.; VALEUR, F.; VIGNA, G. **Intrusion Detection and Correlation Challenges and Solutions**. Santa Clara, USA: Springer-Verlag TELOS, 2004.

LAKKRAJU, K.; YURICK, W.; LEE, A. **NVisionIP: Netflow Visualizations of System State for Security Situational Awareness**. Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC), pp. 65–72, 2004

LAU, S. **The Spinning Cube of Potential Doom**. Commun. ACM, vol. 47, pp. 25–26. 2004.

NORTHCUTT, S.; NOVAK, J. **Network Intrusion Detection.** 3.ed. New Riders Publishing, 2002. 512p.

ONWUBIKO, C.; OWENS, T. J. **Situational Awareness in Computer Network Defense: Principles, Methods and Applications: Principles, Methods and Applications**. Premier reference source. Information Science Reference. 2011.

POPPI, S. **Snort IDMEF output Plug-In**. 2004. Disponível em: <http://sourceforge.net/projects/snort-idmef/>

SHEFFLER, B. The **Design and Theory of Data Visualization Tools and Techniques.**

SIRAJ, M.; HASHIM, S. T. **Modeling Intrusion Alerts using IDMEF Data Model**. Disponível em: <http://comp.utm.my/pars/files/2013/04/Modeling-Intrusion-Alerts-using-IDMEF-Data-Model.pdf>

SNORT. **Snort.Org.** 2015. Disponível em: <https://www.snort.org/>

YIN, X.; YURICK, W.; TREASTER, M.; LI, Y.; LAKKARAJU, K. **Visflowconnect: Netflow visualizations of link relationships for security situational awareness**. Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, VizSEC/DMSEC 04, pp. 26–34, 2004.