FLOVIS: A NETWORK SECURITY VISUALIZATION FRAMEWORK

by

Teryl Taylor

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2009

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "FLOVIS: A NETWORK SECURITY VISUALIZATION FRAMEWORK" by Teryl Taylor in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: April 17, 2009

Supervisors: Dr. John McHugh

Dr. Stephen Brooks

Readers: Dr. Morven Gentleman

Dr. Carrie Gates

Table of Contents

List of Figures

Abstract

Security analysts examine gigabytes of network data on a daily basis looking for signs of intrusive behaviour. Command-line tools such as the System for Internet-Level Knowledge (SiLK) tool suite are helpful but the volume of data makes analysis difficult. We present the FloVis Netflow Visualization Framework, an extensible visualization platform meant to compliment tools such as SiLK for network analysis. Visualization is compelling because it allows the user to view significant portions of data at once and utilize his/her high bandwidth vision and pattern matching abilities for rapid data analysis. FloVis is unique because visualizations are dynamically loaded plugins within the framework, meaning that new visualizations can be added to the system as desired. In this thesis, we discuss the general framework along with three such plugins: FlowBundle, NetBytes Viewer and the SiLK Query Tool. FlowBundle shows connections between hosts on a network using bundling and node aggregation in order to reduce occlusion; NetBytes Viewer provides detailed host volume information per port/protocol over a time period using a 3D impulse graph; and, the SiLK Query Tool is a graphical front-end to the SiLK analysis tools for viewing raw NetFlow records in a tabular form. The system supports drill down and interaction between the different visualizations so that users can see the data in various ways. In addition to describing the existing state of FloVis, the thesis also discusses case studies as well as an informal user study. Finally, a discussion of the future direction of the framework is offered.

Acknowledgements

I would like to thank both my supervisors Dr. John McHugh and Dr. Stephen Brooks for the experience, creativity and input they have brought to the development of this work. Also, I would like to thank Dr. Carrie Gates, Joel Glanfield and Diana Paterson. FloVis would not have been possible without these great team members who put their heart and soul into the project. Much appreciation also goes out to Ron McLeod of TARA for all of his support and resources. I also thank both CA Labs and NSERC for their support in the research initiative.

Last but not least, I would like to thank my family as well as Anh and Nhi. They helped me find a balance in my life during the long hours of doing work and school at the same time. I would be lost without them.

This material is based upon work supported by the Department of Homeland Security under Contract No. N66001-08-C-2032. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Homeland Security.

Chapter 1

Introduction

A network security administrator is charged with monitoring his/her network and looking for signs of behaviour from intruders who are attempting to compromise and control a vulnerable host. Administrators may take a network-based approach to intrusion detection by investigating the traffic on the network. For small networks of a few hosts, packet sniffing software like Wireshark[1] is useful for detecting traffic anomalies. However, once networks reach hundreds or thousands of nodes[2], traffic levels can reach gigabytes per day; thus, more complex techniques for intrusion detection are required. To deal with this issue, automated intrusion detection systems (IDSs) were created. IDSs come in two flavours: anomaly and signature-based systems [7, p. 149]. Anomaly-based systems use statistical and machine learning techniques to learn the normal activity of the network and to alert the administrator when activity deviates from the norm. The problem with this type of system is that it is fraught with high percentages of false positive alarms (warnings on legitimate traffic) and false negatives (missed intrusive behaviour). In signature-based systems, known malicious traffic patterns are converted into a set of signatures or rules. These rules are compared with new traffic to detect intrusive behaviour. The problem with this approach is that only intrusions which already have signatures are detected and existing intrusions can be slightly modified to elude such systems. As a result, research has started in other areas.

The security community has begun to evaluate information visualization as a third complementary approach to intrusion detection. The idea behind information visualization is to transform abstract data into easily understandable graphics to support decision making [7, p. 1]. It is appealing because it takes advantage of human vision which is a high-bandwidth parallel process that excels at rapidly locating and identifying patterns [7, p. 2]. With visualization, an analyst can take gigabytes of

[1]`http://www.wireshark.org`

[2]In this context a node refers to a host.

network data and create an image or mental model of the data. This allows him to recognize patterns of intrusive behaviour much more quickly than looking through raw data. The real challenge in visualization is creating images that are more than simply attractive. They must provide insight into the data without confusing the user. The images must also have a limited purpose (or scope), otherwise they might show too much information which could result in confusing rather than helping the administrator [7, p. xvii].

Visualization is also useful for monitoring normal network behaviours. For example, a network might experience high traffic volumes which could cause the slow down of key services such as web or email. With network visualization, an administrator could pinpoint the exact cause of a slow down and make efforts to correct the problem rather than combing through text-based output from analysis tools.

However, research in security visualizations is still in its infancy. There are relatively few full fledged visualization systems for security as most applications provide only one view of the data (see Section 2.2 for details). Also, most visualizations focus on small networks and do not provide much in the way of user interaction. Lastly, many of the key ideas that have taken hold in other areas of visualization have not yet been applied to security visualization (e.g. zooming, drill down, and occlusion reduction techniques).

In this thesis, we describe a new visualization system called FloVis with the main goal of providing security administrators insight into what is occurring on their network. With FloVis, the philosophy is the following:

1. Provide several different views for the user. These views will focus on host and network traffic activity including:

    - Where is traffic flowing to/from on the network?
    - How much data is flowing?
    - What type of traffic is flowing (e.g. protocols)?
    - What ports are used?
    - What are the traffic patterns over time?
    - Who is talking to members on the network that should not be?

2. Apply new visualization techniques to security in order to minimize some of the issues common to current security visualizations.

3. Create a fully integrated solution so that data is analyzed and easily imported into the visualizations on demand.

4. Provide a means to move from the visualization to the underlying raw data when more detailed investigation is required.

5. Provide an extensible framework for rapid visualization development, integration and customization.

Point number one is key to the approach. One visualization can rarely offer the security administrator all the information she requires [10]. With FloVis, a system of visualizations was created, with each visualization giving a different view of the data and each linked to one another through drill down and interactive features. The idea is that if one visualization does not provide the required insights then another might. This also ensures that no single view becomes overloaded with information to confuse the user. In this thesis, two visualizations are presented. The first is the FlowBundle visualization, which shows how networks and hosts interact with one another through traffic flow connections. In this view, a new bundling technique [17] is utilized to tackle occlusion issues that plague current security visualizations. FlowBundle also has several interactive features that allow the user to manipulate the data and drill down to find interesting details about the network. The second application, called NetBytes Viewer [38], is a network entity-based visualization (where entities may be individual hosts or networks) that shows flow, byte, or packet volumes for these entities over time using a 3D impulse graph. Data can be categorized by port or protocol in order to detect patterns of intrusive behaviour. Large spikes in data on a specific port or traffic on unusual ports might indicate intrusive activity.

With the creation of the FloVis framework, an extensible open source platform and windowing system was developed that allows for the integration of multiple visualizations into one application. With FloVis, security analysts are able to mix and match visualizations that they find beneficial to their analysis and security and visualization researchers can continue to develop new ones to suit their needs. Furthermore, security visualization researchers are able to focus their efforts on the design and development of visualizations rather than writing application-level windowing code. In the end, it is hoped that FloVis will provide the user with a suite of useful visualizations to help provide insight into one's network.

The remainder of this thesis will proceed as follows. In Chapter 2, we will investigate the strengths and weaknesses of current security visualizations, as well as some related research in the broader area of visualization. We will also briefly discuss a network analysis toolkit called SiLK, which allows users to collect NetFlow level statistics on their network. It provides the basis for the data component of our visualizations.

Chapter 3 discusses the FloVis framework along with the two visualizations noted earlier: FlowBundle and NetBytes Viewer. In this section, we also present a graphical user interface for the SiLK tools called the SiLK Query Tool.

Chapter 4 investigates some of the implementation considerations for the visualizations. Most notably, we discuss how raw SiLK data is processed and stored for easier access by both NetBytes Viewer and FlowBundle. With the volume of data produced by large networks on a daily basis, data storage and access is of extreme importance to this area.

We evaluate the FloVis framework using two different methods (Chapter 5). First, we outline three different case studies where the FloVis visualizations were used to find interesting patterns of intrusions using a private network dataset. Second, we summarize the user feedback we received when conducting an informal user study with a dozen security analysts at Flocon[3] 2009. It was important to select trained security analysts for evaluating the application because these are the people that will ultimately use FloVis on a day-to-day basis. The informal user study will help to convey the usefulness of FloVis and at the same time provide some key ideas for a future direction for the product.

We close with our conclusions and suggest future work for FloVis. Proposals for new visualizations, as well as updates to current ones, are presented in Chapter 6.

[3]http://www.cert.org/flocon/

Chapter 2

Background

This chapter provides the necessary background information required to provide context for the rest of the thesis. First, we begin with a description of the data and analysis used by the FloVis framework. Next, we provide a discussion of related work in order to differentiate FloVis from existing research.

2.1 NetFlow and the SiLK Analysis Tools

Before discussing FloVis in detail, we must first explain NetFlow and the SiLK Tools[1] which provide the basis for the underlying data used in the FloVis framework. NetFlow is a system developed by Cisco for collecting statistics about network traffic for management and billing purposes. These statistics describe the interactions between hosts across a network border. NetFlow is a proprietary protocol but is open and supported by more than just Cisco routers. Typically, NetFlow-enabled switches or routers report statistics on a per interface basis. NetFlow records are generated by the router and exported to a collector over UDP where they are stored for further processing as shown in Figure 2.1.

Each NetFlow record includes the following information [18]:

1. Source and destination IP addresses.
2. Input and output interface numbers.
3. Source and destination ports for TCP/UDP (message type and code for ICMP).
4. Protocol number.
5. First and last timestamps that were part of the flow.

[1]The SiLK Tools are available as open source under the GPL from `http://tools.netsa.cert.org/silk/index.html`.

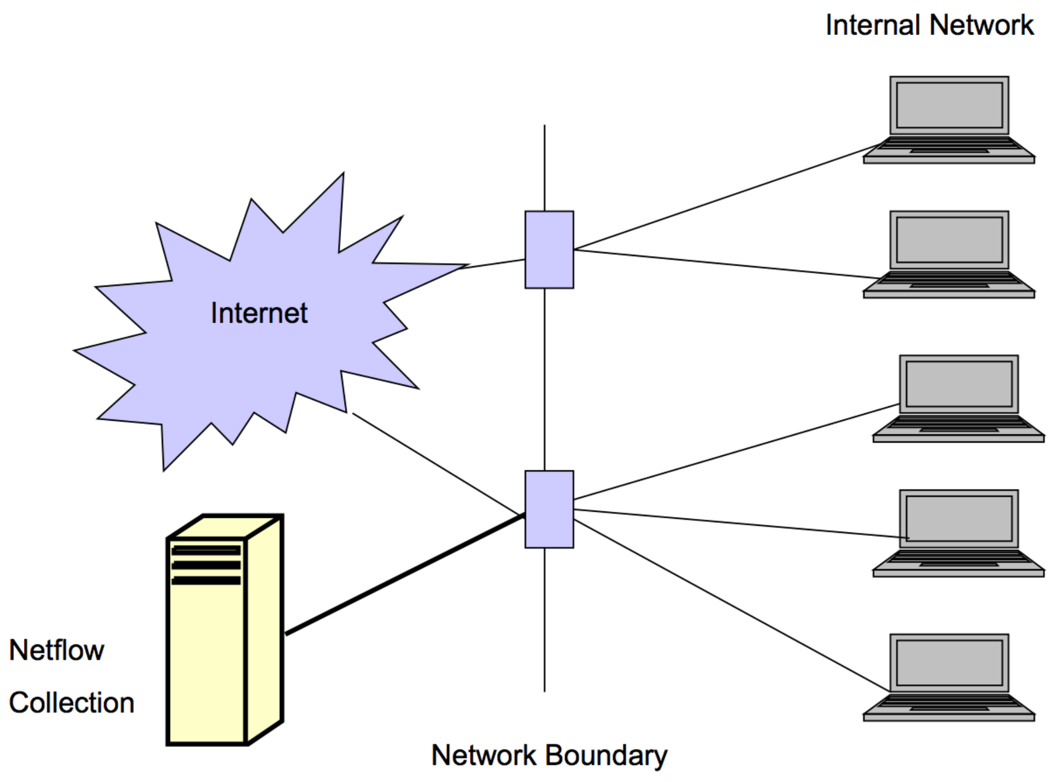


Figure 2.1: A typical setup for NetFlow collection on a network border.

6. Number of packets in the flow.

7. Number of bytes in the flow.

8. TCP flags for all packets in flows from TCP sources.

Additional information may also be collected including routing information, *e.g. next hop IP* etc.

NetFlow records represent unidirectional flow information between two hosts and are somewhere between packet header and session level information. They do not contain packet payloads as do other passive tap applications (*e.g.*, `tcpdump`). NetFlow data records are also more compact than `tcpdump` records. Although information is lost due to missing payload information, the more compact NetFlow records require significantly less disk space and processing time and avoid privacy issues associated with storing packet payloads.

The CERT Network Situational Awareness Team (CERT NetSA) developed the

SiLK tool-set to support the analysis of large scale NetFlow data. SiLK contains command line tools that collect a modified form of NetFlow records and pack them in binary flat files organized by time and partitioned by service. The flow data collected by CERT adds additional information that is specific to performing security analysis (such as keeping the flag combination for the first packet in a flow), and does not contain the routing-specific information contained in NetFlow records. SiLK also contains a set of analysis tools that provide a number of query operations on the dataset, including filtering and statistical analysis.

One of the features of the SiLK analysis suite is the ability to generate sets (*e.g.*, of IP addresses or ports). Sets are a compact way to describe and group traffic observed on the Internet [28]. For example, the user may want to analyze the number of hosts in a subnet that were active during a given time period in order to characterize host behaviour. Furthermore, he/she also might want to find the set of internal hosts that were communicating with a specific external host on a suspicious port, or the set of external hosts that were scanning the internal network. SiLK computes these sets by analyzing the flow database using the appropriate filters.

Another feature of SiLK — one that provides the basis for most of the analyses used in this thesis — is the ability to generate bags. A bag is a counted set, or multiset, in which the number of occurrences of each member of a basis set is recorded [28]. For instance, the user may want to look at the number of bytes per hour, or the number of flows or packets per port, host, or subnet during a specific time period. Activity on suspicious ports or large traffic spikes on regularly used ports could indicate malicious activity. Bags can be indexed by IP address, protocol number, ports, as well as other fields contained in a flow record. They can also be indexed by composite values of up to 32 bits that are taken from any two scalar fields of a flow record.[2]

For most of the visualizations, the SiLK records were processed into bags based on host or port/protocol characteristics. Detailed bags were created for each of the active hosts. Most of the data that drives the visualization is derived from a time series of hourly bags.

[2]This requires the `rwconnectbagtool`, which the FloVis team is contributing to the tool set (see Section 4.3.1 for details).

2.2 Related Work

In this section we examine various visualization techniques that have been applied to the area of network security; we compare and contrast the FloVis approach with each of these as appropriate. Additionally, we present several visualization techniques that have been applied in other domains (*e.g.*, science, finance, geography) and that have been adapted for FloVis.

2.2.1 Security Visualizations

The following subsection gives an overview of the security visualization domain by describing several existing visualizations and their relationship to FloVis. In the first portion of the section, visualizations that focus on the interactions between hosts on a network are discussed. These types of visualizations convey who is talking to whom on a network. Next, entity-based (host or subnet) visualizations are investigated. These types of visualizations focus on the behaviour of a single host or subnet. Finally, some miscellaneous visualizations are explored to provide a broad overview of the network security visualization domain.

One technique used in security visualization is the representation of network communications in a *parallel coordinate system* [7, p. 44]. Each variable (or dimension) to be visualized is represented by a single vertical axis. Values for each variable are mapped to locations on an axis and relationships between values are represented by lines. Figure 2.2 shows an example of a parallel coordinate system with three variables. With respect to networking, an example use of such a graph involves representing source IP addresses of a network on one axis *(A)*, while representing destination IP addresses on another *(B)*. Connections between source and destination hosts are represented using lines drawn between parallel axes.

One application that utilizes parallel axes (in two dimensions) is VisFlowConnect [42], which visualizes NetFlow connections between computers across a network border. An axis represents either an internal or external network. Each point on an axis represents an IP address of a machine or domain and connections between points on the axes represent network connections and data flow. VisFlowConnect depicts connections over time with animation and colour, i.e., as a connection occurs a line

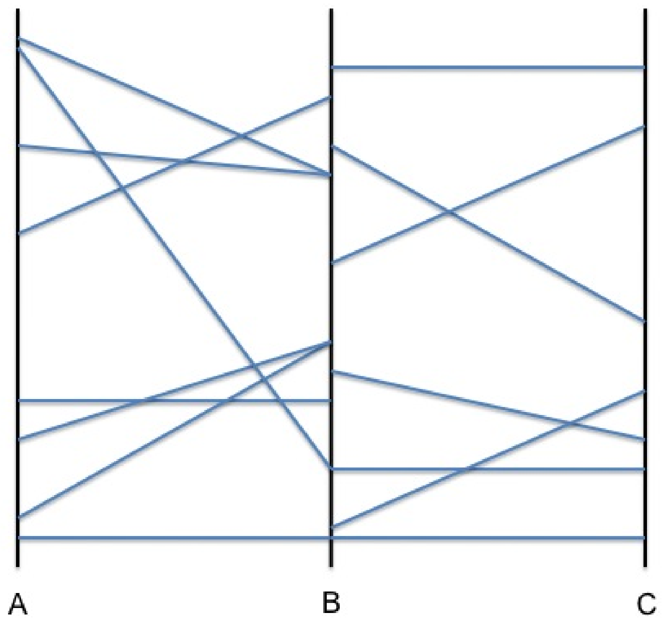


Figure 2.2: A simple parallel coordinate system with three variables.

is drawn in a dark colour that fades over time; hence, animation provides temporal context for data flow. VisFlowConnect also has controls to stop, rewind and replay the connection history.

VisFlowConnect provides the option to filter by port, protocol, transfer rate, and packet size, all of which allow the user to exclude hosts that are deemed uninteresting. VisFlowConnect shows individual-host statistics in a textual format. Statistics include the total number of bytes transferred to and from a host in the current time window, and total bytes transferred per host.

One disadvantage of VisFlowConnect is that IP connections are represented by straight lines which tend to overlap. Under high traffic volumes (*e.g.*, network scans) occlusion causes the lines to become indistinguishable [8]. This can make analysis difficult unless a substantial number of "uninteresting" hosts are removed from the visualization. The application is also limited in its ability to drill down and examine individual entities (hosts and subnets) as it only provides a connection oriented view of the data. Additionally, it is difficult to distinguish traffic volumes that may indicate anomalous behavior. FloVis addresses these problems by utilizing connection bundling in such a way as to prevent occlusion. It provides several views of the data at both the entity and connection levels, allowing an analyst to drill down and investigate data flows. The motivation for providing multiple views is that relevant patterns in the data may be more easily detected when compared to a single visualization, which may include too much information and potentially obscure interesting patterns.

Portall [11] is an interactive tool that visualizes the correlation of host process communication with network traffic; that is, it shows interprocess communication between hosts on a network. Portall also uses parallel axes (2D), in which one side of the application contains a column of client IP addresses while the other side shows server IP addresses. Each IP address points to a set of boxes in an adjacent column which list a set of processes and their corresponding ports. Client-server connections are displayed using straight lines from one axis to another while transmission volumes are shown with tooltips. Tooltips are informational boxes that appear when a mouse cursor hovers over a point of interest. Colour is used to represent Transmission Control Protocol (TCP) connection states. Time is represented using animation, with processes and connections appearing and disappearing over time in a similar manner as VisFlowConnect. To the author's knowledge, Portall is the first visualization tool that attempts to correlate traffic flow to host processes. However, Portall does not scale to large networks and requires a central database to store process information.

VISUAL [3] is a two dimensional (2D) visualization application that shows the traffic between an internal network (of up to 2500 hosts) and an external network (of up to 10000 hosts) using *tcpdump*[3] data. Internal hosts are represented as small cells in a square grid in the center of the screen. External hosts are represented as squares placed in the area surrounding the internal grid. The size of each cell denotes the amount of IP activity of a host and lines in the square are used to denote port traffic. Straight lines between internal and external cells are used to denote traffic flow while the grid colour denotes internal communication between computers. VISUAL allows interactive filtering of specific computers to reduce screen clutter and uses animation to show changes in network flow data over time. Detailed information of individual hosts is available in textual form and includes: the host IP address, the IP addresses of all computers with which it has communicated, ports used and percentage of overall traffic. Although VISUAL suffers from the same occlusion issues as VisFlowConnect

[3]See http://www.tcpdump.org/

under high traffic volumes, it does improve upon VisFlowConnect by incorporating volume data as well as flow data. FloVis goes beyond both by providing a full range of drill down facilities to allow the user to see volume information for each host or subnet on a per protocol/port basis.

The Radial Traffic Analyzer (RTA) [21] uses a hierarchical radial layout (2D) to show the relationship between source and destination hosts and ports. Each attribute becomes a concentric ring divided into sectors. Sector size is based on volume or number of connections. The most important attribute (as decided by the user) is placed in the innermost ring. As more data points are plotted on the circle, sectors become too small for labeling and tooltips are used on mouse overs. This is a major disadvantage for RTA. There is a limited amount of space for the sectors and as more data is visualized on the diagram, the sectors become so small that it is difficult to analyze the output. FlowBundle overcomes these problems by enforcing an upper limit on the number of data points in the view at one time and allowing drill down to increase resolution at the expense of scope.

Scatter plots are also used to visualize two dimensional data; two example security tools are NVisionIP [23] and Portvis [30]. Both show the current state of the network as a whole as well as individual host activity.

NVisionIP presents a visual representation of NetFlow traffic for a /16 network on a single screen. The overview screen has horizontal and vertical axes; all subnets of the network are listed along the horizontal axis while the hosts in each subnet are listed on the vertical axis. Each host is coloured based on a characteristic of interest, *e.g.*, traffic volume, number of flows, or flows on a particular port. Animation is used to show traffic collected over a given time period. Users can select a region of the overview screen to launch another window that provides more detailed information about hosts in the selected region. Each host is represented by two bar charts. One chart displays the traffic on a number of well known ports while the other shows traffic on all other ports. Colour is used to differentiate traffic on different ports to make it easier to compare flows of interest. NVisionIP also has a host view which uses 2D bar charts to display byte and flow counts for all protocols and ports. Although the drill down mechanism allows users to view data on individual hosts, it does not visualize patterns in the data over time on a single screen. The FloVis NetBytes

Viewer (see Section 3.3) integrates a time axis, the goal of which is to aid an analyst in discovering anomalous data patterns without having to rely on short term memory. While NVisionIP displays host traffic and traffic volumes, it does not show host-to-host communications. This information is visualized using FlowBundle in FloVis.

Another visualization tool, Portvis [30], incorporates three different displays to view TCP port traffic. The first visualization is a 2D grid where rows along the vertical axis represent units of time, and each column along the horizontal axis represents a range of 2048 ports. Colour in each column is used to represent the level of activity on the ports during a particular time interval. The interval to be displayed on the main visualization can be selected by the user. The main visualization contains a 256×256 grid where each point represents one of the 65536 ports. The location of the port on the grid is determined by breaking the port number into a two-byte (x, y) location. The grid can be magnified in specific areas to provide more detailed information about specific ports. The tool contains a port display which displays information on the attributes of a selected port. The 3D display is a set of line graphs that depicts time on the horizontal axis and traffic volume on the vertical. Although Portvis visualizes port traffic, time-dependent patterns are defined very coarsely (*i.e.*, 2048 port buckets) which could make detecting temporal patterns difficult. There may also be issues of information overload especially under conditions involving multiple sessions and multiple services (which causes the port grid to become saturated with many colours). FloVis improves upon this shortcoming by having NetBytes Viewer display time as a third dimension. Portvis focuses on temporal and volume based port traffic but does not visualize host interactions on those ports.

Flamingo [31] is another visualization tool that uses NetFlow data (NetFlows). It is composed of: 1) a server which collects and processes NetFlows, and 2) a client which displays the processed data using a number of different visualizations. The first visualization uses a graph to depict network traffic based on IP addresses. It uses a quad-tree based algorithm to represent all the IP addresses and subnets in a square structure (on the x, y-plane), and counts the NetFlows (based on prefixes) and represents the flow counts by bar height on the z-plane. The bar thickness represents the size of the network and colour is used to represent each prefix. The user can zoom in on prefixes to obtain a more detailed view and can choose to examine either source or destination IPs. Flamingo uses a similar graph to represent port traffic; however, it replicates the IP network bars at different heights to represent per-port traffic for the IP address range. A third visualization is used to represent the connections between source and destination IPs (or subnets). One square at the end of a cube is organized in the quad tree IP representation and represents the source IP addresses; the other end of the cube represents the destination IP addresses. Connections are shown as straight lines from one end of the cube to the other. There is also a visualization that combines all of the above elements. It uses a cube with the bottom of the cube representing the source/destination IP address space and the source/destination ports are represented on the z-axis. Lines are used to connect a source IP/port pair with its corresponding destination IP/port pair. Line thickness is used to represent flow volume while colour is used to show flow direction. Flamingo differs from the visualizations surveyed above in that it uses a third dimension to display more data. The 3D cube allows the visualization to show IP addresses as a quad-tree and the actual data in the third dimension. This allows it to display more IP addresses than the other visualizations; however, there are disadvantages. The quad tree makes it difficult to locate and label a specific IP address. The application also appears to suffer from depth perception issues. The straight line connections still create occlusions as mentioned above. On the positive side, Flamingo recognizes the importance of using multiple visualizations to show both connections and data volume. This approach is used in FloVis. As previously noted, any one representation may offer a more beneficial view of an event of interest.

Another visualization that utilizes a 2D style IP matrix is StarMine [16]. StarMine takes two 2D visualizations — a logical visualization (the IP matrix in the vertical orientation) and a geographical map visualization (horizontal orientation) — and links them together using connection links, thus resulting in a 3D graph. The application visualizes attacks recorded by Snort[4], an open source intrusion detection system, and maps IP addresses to their geophysical locations to allow a view of the locations of attacks. The vertical axis displays histograms at locations on the map to depict the number of attacks at a location. The IP matrix makes it difficult to find a specific IP address; however, this technique could be valuable in linking related 2D visualizations.

[4]See http://www.snort.org/

A generic version of StarMine, called VisLink [6], is built around the concept of linking any two 2D visualizations.

The “Spinning Cube of Potential Doom” [24] is another network visualization tool. It was built on top of the Bro[5] intrusion detection system and displays a 3D cube which may be rotated interactively. As described in [24], “Each axis represents a different component of a TCP connection: X is the local IP address space; Z is the global IP addresses space; and Y is the port numbers used in connections to locate services and coordinate communication.” TCP connections are displayed as individual dots and colour is used to distinguish successful versus unsuccessful connections. Time is represented through the use of animation and the cube can be rotated to eliminate occlusion. The major difficulty with this visualization is that it shows individual connections as dots in a 3D cube which can make it difficult to identify the particular traffic connection being represented (which is mainly a perception issue). The Spinning Cube provides an animated pattern that may be useful for detecting intrusive behavior, but it provides few features to allow individual machines to be analyzed. FloVis provides many interactive features including zooming, rotating, and data selection, all in an effort to give the user additional analysis capabilities and eliminate both perception and occlusion issues.

The Time-Based Network Traffic Visualizer (TNV) [15] uses several visualization techniques. It takes a *focus + context* approach and provides the user with historical context so that intrusions can be detected by recognizing changes in activity patterns. The *focus + context* technique was originally proposed by George Furnas [13], and has been utilized in several visualizations (*e.g.*, Table Lens [34], Perspective Wall [26], Document Lens [36]). It provides more detail and allocates more space for the most important information while surrounding the detail with contextual information, which is distorted in space to reduce clutter. TNV takes a tabular approach similar to TableLens, where columns represent time periods and rows represent hosts. Columns that have focus are wider and provide more detailed information (*e.g.*, host connection information and port activity). Conversely, contextual columns are more narrow and relay volume information for less interesting hosts. TNV provides zooming and exploration features to allow the user to look at

[5]See http://www.bro-ids.org/

different areas in more detail. It provides an overview time line mechanism, as well as a table to show raw data. The visualization emphasizes the importance of internal hosts by using a bold-faced font. To save space, port activity (per host) is aggregated and volumes are shown as a histogram. TNV emphasizes the importance of using historical context to identify normal patterns and subsequently observing how malicious activity affects those patterns. This is an important aspect of security analysis and is discussed further by Goodall [14]. TNV demonstrates that there is a trade off between the number of hosts displayed and the detail provided for each one; hence, TNV does not scale to large networks. The contextual approach of TNV provides situational awareness; FloVis similarly applies different approaches to providing contextual information. Like TNV, FloVis allows the user to retrieve the underlying data for further analysis. Unlike TNV, FloVis scales to larger networks by striking a different balance between screen real-estate and hosts displayed.

Isis [33] takes a somewhat similar approach to TNV in that it uses *timelines* to give historical context to data. It contains two linked visualizations: a timeline and an event plot. In both views, IP addresses are categorical values along the vertical axis while time is represented along the horizontal axis. Timelines show aggregate values over all events using bar graphs at each time interval, while the event plot shows the patterns of individual events using glyphs. The visualization operates on the premise of a focus IP address (i.e. an IP address of interest). From that focus IP address, the user can pivot to other related IP addresses or filter data using SQL inside of the visualization. Each change to the SQL query launches a new timeline underneath the existing query to provide a historical query context. Incorporating the SQL language directly into the visualization provides an interface for data manipulation. Isis would be an interesting complementary visualization to the other visualizations in the FloVis framework.

Another aspect of networking that is considered in many visualizations is the hierarchical nature of IP addresses. Tree-based visualizations as applied to network monitoring and security include *TreeMaps* and/or Interactive Hierarchical Network Maps (HNMaps) [27]. A TreeMap maps hierarchical information to a 2D display of rectangles designed to fill the available screen real-estate [20]. The rectangular bounding boxes represent nodes on a tree. Boxes placed inside other boxes represent deeper nodes (*i.e.*, boxes within other boxes are considered child nodes). Box size can be used to emphasize the importance of nodes while colour may be used to encode another dimension. TreeMaps have been used for network visualization because they can concisely display thousands or even millions of IP addresses. Due to the hierarchical nature of the maps, IP addresses within the same subnet are placed near each other. Mansmann [27] gives a high level view of the traffic on the Internet by creating a hierarchy for IP addresses and allowing the user to drill down. At the highest level, IP addresses are grouped by country. Each country is represented by a rectangle whose size is proportional to the number of IP addresses within that country. Countries are arranged as they appear on a map with the omission of landmarks and oceans. Users may click on a country to drill down and view the autonomous systems (AS) for that particular country. Users may then click on an AS to view a TreeMap of the IP prefixes that are inside the AS. Colour is used to represent traffic activity for each prefix. Such an application may be useful for showing the overall propagation of worms or hot spots of activity on a network; however, it may be more beneficial to combine this approach with other visualizations to improve the detection of malicious activity. Another approach, somewhat similar to Treemaps, is *Hilbert Curves* [19]. Hilbert Curves can be used in network security to map the entire IP space to a set of points on a visualization space whereby IP addresses that are close together numerically are close together spatially. A Hilbert Curve is able to show millions of IP addresses on the screen at once, but offers no detailed information about any particular IP address. Labeling and drill down are unavailable on these types of graphs.

IDS Rainstorm [1] was designed to visualize IDS alerts for the Stealthwatch anomaly-based IDS system. The main visualization divides the display into a set of columns representing a contiguous set of IP addresses. The horizontal portion of each column represents a 24 hour period of alerts. A row of coloured dots represent the total number of alarms for the IP addresses at a particular instant in time. The user may use a mouse to highlight a range of IP addresses. Upon clicking on a selection, a secondary window is launched with an enlarged view. The selected IP addresses appear on a vertical axis on the right side of the window. Time appears on a horizontal axis and the alarms appear as large coloured glyphs. This type of activity illustrates how a drill down mechanism may be used to view specific data in more detail. The NetBytes Viewer uses similar drill down techniques to launch 2D views of multiset data as an attempt to help the user gain additional insight into network traffic patterns.

Komlodi *et al.* [22] describe a glyph-based visualization that can be customized by mapping data variables to visual glyph attributes (*e.g.*, colour, size, position, etc.). Visualizations can be rendered in both 2D and 3D forms. This user centric approach demonstrates the need for flexibility when developing visualization tools. Flexibility is an important component for FloVis. The framework allows visualizations to be mixed and matched giving the user freedom to customize her environment. Furthermore, the FlowBundle visualization enables the user to show the relationship between any two data attribute pairs (*e.g.*, source IP, destination IP, source port, destination port, protocol) in a NetFlow record.

ViAssist [10] is the closest application framework to FloVis. It incorporates simple visualization paradigms such as bar charts, parallel coordinate systems and node-link graphs into a single application allowing for transitions between views. It focuses mainly on NetFlow traffic but can integrate data from other sources such as IDS logs. Although not extensible at the moment, the ViAssist developers are heading in that direction. Furthermore, the application has some interesting interactive features such as annotation and E-diary to support group collaboration. In contrast to ViAssist, FloVis is currently extensible, open source and platform independent - ViAssist is a Windows-based application and is proprietary.

2.2.2 Visualization Techniques Applied to FloVis

It is always beneficial to investigate outside security visualization to find useful visual paradigms that can be applied to the area. In this section, novel visualization techniques from other areas are discussed and related to FloVis.

ExtraVis [9] visualizes function call sequences (execution traces) within an application. It is intended to aid a maintenance programmer in learning the structure of a program rapidly. It provides two views; the first is a view of a circle with a program's structural entities arranged around the outside of the circle. The calling relations between entities are displayed using edge bundle connections [17]. Edge bundling bends adjacent edges in a hierarchical structure together thereby reducing visual clutter. It allows the user to discern the general path of the grouped edges without worrying about the edges fanning out which causes occlusion. ExtraVis' circular view provides a snapshot of the data that corresponds to the part of the execution trace that is currently under review. The thickness of a connection indicates the number of calls between two entities, while a colour gradient is used to indicate direction. Colour intensity is used to indicate call order. The second view is a sequence view that gives an overview of the execution trace. The systems structure is shown along the horizontal axis while time is indicated along the vertical axis. The application provides a zoom-in feature, which allows the user to focus on a section of the trace. This circular visualization provided the inspiration for the FlowBundle application (see Section 3.2). FlowBundle makes use of basis-spline (B-spline) bundles in an attempt to reduce visual clutter when representing connections between hosts (or subnets) on a network. The technique was applied along with some other key features such as drill down, distortion and node aggregation.

There is one security visualization that has applied bundling to their approach - NFlowVis [12]. NFlowVis combines bundling with TreeMaps to create a visualization somewhat similar to VISUAL [3]. However, links are bundled and a TreeMap is used instead of a matrix for internal hosts. Though a novel approach, the benefits of the bundling technique are offset by the negatives of the TreeMap for this application. Thousands of hosts are spatially close to one another and with high traffic levels the visualization becomes extremely cluttered.

Semantic substrates [2] are spatial templates for networks. Nodes are grouped into regions and are organized by one or more attributes as specified by a user. Groups of nodes may be aggregated to reduce screen clutter. Other features include filtering and generating details on demand. Grouped nodes are organized into tabular form. As a node is aggregated, the size of the aggregate is increased to give the user an idea of the relative number of active nodes within the aggregation. Elements of this technique (*i.e.*, node aggregation) were applied to the FlowBundle application to represent networks with large numbers of nodes.

Chapter 3

Approach

As evident in the related work, research in the area of security visualization focuses on modeling network data using as many data points as possible. Scatterplots, parallel coordinate systems, 3D-cubes, quad-graphs, Hilbert Curves, TreeMaps, and matrices are all visual structures[1] being used and are attempts to display data points at the pixel or near pixel level. They allow for millions of data points to be displayed on the screen at once in order to convey information about the millions of entities (hosts, networks, ports, protocols) comprising a network. The advantage of these types of visualizations is that they provide an overview of the traffic on the entire network. This is important because some networks can have billions of nodes. However, with these advantages comes some significant disadvantages. They are as follows:

- Information is displayed about millions of items at once, but nothing is displayed in any detail. It becomes almost impossible to obtain deeper information about individual hosts/entities because data points are too small to highlight and manipulate. Drilling down can often provide more information about the root causes of happenings on the network.

- Effective labeling is almost impossible because data points are too small and too close together. Even if you find a pattern of malicious behaviour you may not be able to identify exactly which host is causing the problem because there is no way to identify the host in the visualization.

- With millions of data points users can experience information overload.

- Occlusion is an issue especially for connection oriented graphs which rely on lines to connect data points. This makes the visualizations themselves susceptible to attack as hackers can create intrusions that cause occlusion and render visualizations useless [8].

[1] A visual structure is a visual format for rendering data points.

Another aspect of security visualization research is simply that there are few visualization systems. In most cases, the research is limited to a single visualization or a set of one or two visualizations focused on some single attribute of the network (*e.g.* host to host connection, traffic volume.). There is no way to tie these visualizations together giving the security analyst a suite of tools for network analysis.

The paradigm presented in this thesis attempts to overcome some of these key shortcomings of existing security visualizations by applying the Ben Shneiderman directive: "Overview first, zoom and filter, details on demand" [37]. The idea is to provide the user with a set of high level network visualizations giving an overview of the entire network organization, and then allow him to drill down (through zooming and/or filtering) on smaller portions of the network to obtain more detailed information. Further, the approach provides a set of visualizations that enable the user to view the data in different forms so that if one visualization cannot show an interesting pattern then another one might. Each visualization has its own set of interactive and drill down features to provide more detail.

There are a number of advantages to this approach. First, providing a high level overview and then drilling down on certain portions of the network limits the number of data points displayed to the user so as not to overwhelm or confuse her. Further, through overview and drill down, the approach is able to eliminate issues of occlusion and reduce the susceptibility of the system to visualization attacks. Next, by utilizing high-level overview visualizations, data aggregation is used to reduce the overall amount of data visualized. Upon drill down, a subset of the data is retrieved in more detail for further visualization. This is important in large networks where billions of flows are collected per day. It is not feasible to process all the data at once; therefore, aggregating and partitioning the data is needed, as well as allowing the user to decide which portions of the network she wants to view in more detail.

To facilitate this, the FloVis framework was developed as a generic extensible platform designed to enable the integration of visualizations into one common application. Visualizations are developed as *plugins* to the framework and are loaded at runtime. The framework is designed to be customizable such that a user can choose which visualizations she wants to use in her network analysis. Provisions have also been made to allow visualizations to communicate with one another through a framework managed event handler. This enables one visualization to launch another using the same underlying data. Furthermore, it allows visualizations to focus on visualizing different aspects of the data so that each can provide different insights into the same data.

The primary goals of the framework are to:

1. Promote visualization research and design through rapid visualization development and integration.

2. Reduce security analysts' network and intrusion investigation time.

Given the primary mission to research visualizations which provide insight into a security analysts network, the FloVis framework is a critical facilitator for minimizing visualization prototype development time and increasing time for visualization evaluation and use.

In the following subsections the FloVis framework is discussed in more detail and two interactive visualizations are subsequently described: FlowBundle and NetBytes Viewer. FlowBundle is a connection-oriented visualization that captures interactions between host and subnets by grouping related flows as bundles. Its primary aim is to help eliminate some of the key issues plaguing current connection-oriented visualizations by incorporating some novel visualization approaches. NetBytes is an entity-based viewer that provides a detailed inspection of volume traffic for a host or subnet over time. This allows the analyst to detect changes in behaviour that manifest from unusual port or protocol usage, and/or traffic volume.

3.1 The FloVis Framework

The FloVis framework [39] is both a toolkit and a windowing system designed to facilitate the rapid development and integration of visualizations into a single application. It benefits the analyst because it provides a common architecture where he can select visualizations that he finds most beneficial in his analysis. The framework is extensible so that the analyst can incorporate new visualizations as they become available. In terms of the visualization researcher, the framework handles most of the infrastructure development in a software application, allowing the developer to focus on the new visualization. FloVis also facilitates group research and development as each visualization is componentized and pluggable — meaning that each person can work on his own visualization and the framework handles integration.

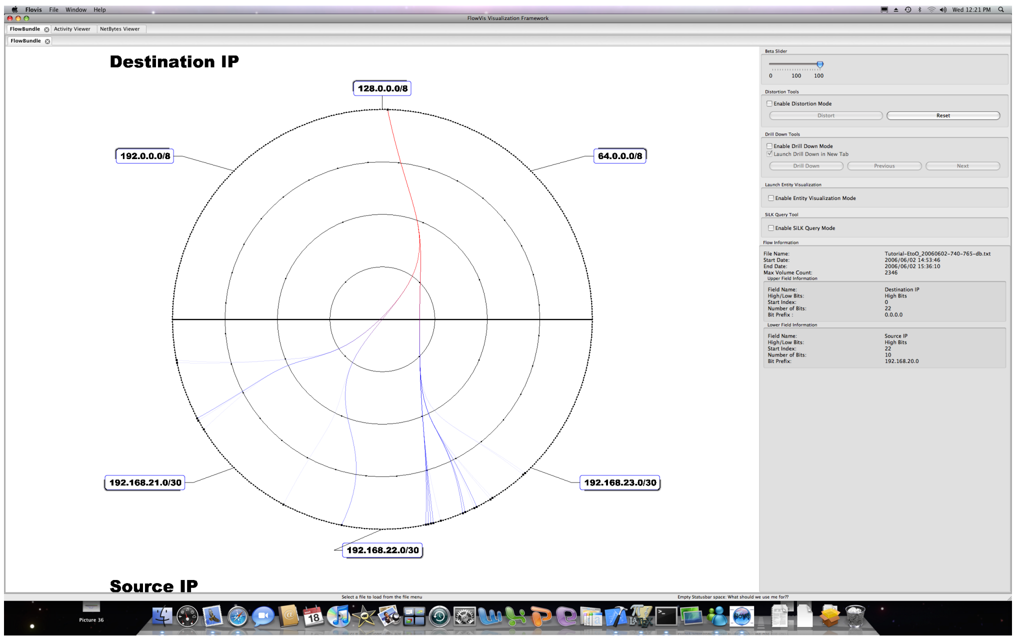
It is also of note that analysts have varying working environments. In visualization, it is important to maximize screen real estate and as such, FloVis was designed with the Internet web browser paradigm in mind. The user decides how many windows she wants open at any time and can launch new visualizations into windows or tabs. Essentially, the user customizes the display layout to meet her needs given her resources. Figure 3.1 shows FloVis in a two monitor scenario. Figure 3.1(a) shows the first monitor which contains a FloVis window. In this case, the user has loaded FlowBundle into the current tab to evaluate the data. She has also loaded the Activity Viewer and the NetBytes Viewer in separate tabs on the window. These visualizations might refer to other hosts of interest and the user can keep the visualizations in separate tabs for future reference. On the second window (shown in Figure 3.1(b)), the user has loaded another NetBytes Viewer visualization. This configuration allows her to view two different (related or not) visualizations at the same time thereby maximizing screen space.

Each visualization is created as a plugin (as shown in Figure 3.2). A plugin is a component that follows a generic API and controls the flow of interaction between the user and the visualization. The framework handles application-level events generated by the user and selects the proper plugin to launch a new (or modify an existing) visualization. These events include the following:

1. Launching a new visualization from the application menu.
2. Launching one visualization from another using interactive controls (*e.g.*, buttons, checkboxes).
3. Modifying an existing visualization using filtering or drill down.

Figure 3.2 shows the three current visualizations developed for the framework: Activity Viewer[2], NetBytes Viewer and FlowBundle [40]. These plugins are able to communicate with one another via a built-in event manager. The event manager contains a mapping between a plugin and an identifier. When an event is generated

[2]The Activity Viewer is a plugin in the FloVis Framework but is outside the scope of this thesis.



- [ ] Enable Distortion Mode
- [ ] Enable Drill Down Mode
- [x] Launch Drill Down in New Tab
- [ ] Enable Entity Visualization Mode
- [ ] Enable SiLK Query Mode

Flow Information

File Name:	Tutorial-EtoO_20060602-740-765-db.txt
Start Date:	2006/06/02 14:53:46
End Date:	2006/06/02 15:36:10
Max Volume Count:	2346

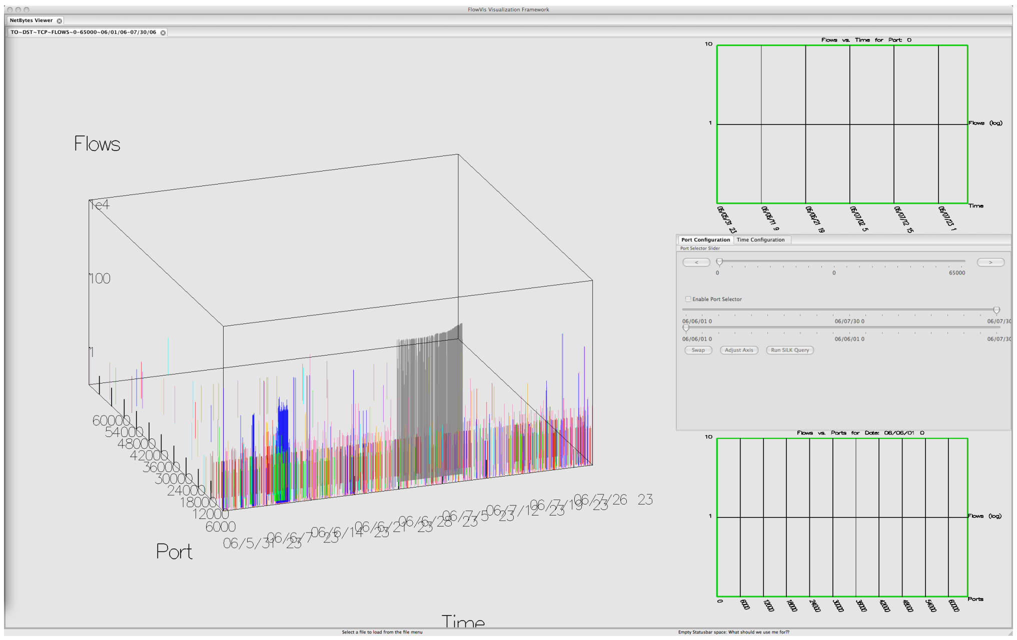
Upper Field Information

Field Name:	Destination IP
High/Low Bits:	High Bits
Start Index:	0
Number of Bits:	22
Bit Prefix :	0.0.0.0

Lower Field Information

Field Name:	Source IP
High/Low Bits:	High Bits
Start Index:	22
Number of Bits:	10
Bit Prefix:	192.168.20.0

(a) Window 1 shows a tab with FlowBundle. See also tabs for NetBytes and Activity Viewers.



- [ ] Enable Port Selector

(b) Window 2 shows a tab with NetBytes Viewer.

Figure 3.1: FloVis framework loaded on a system with two monitors. One window is open on each monitor.

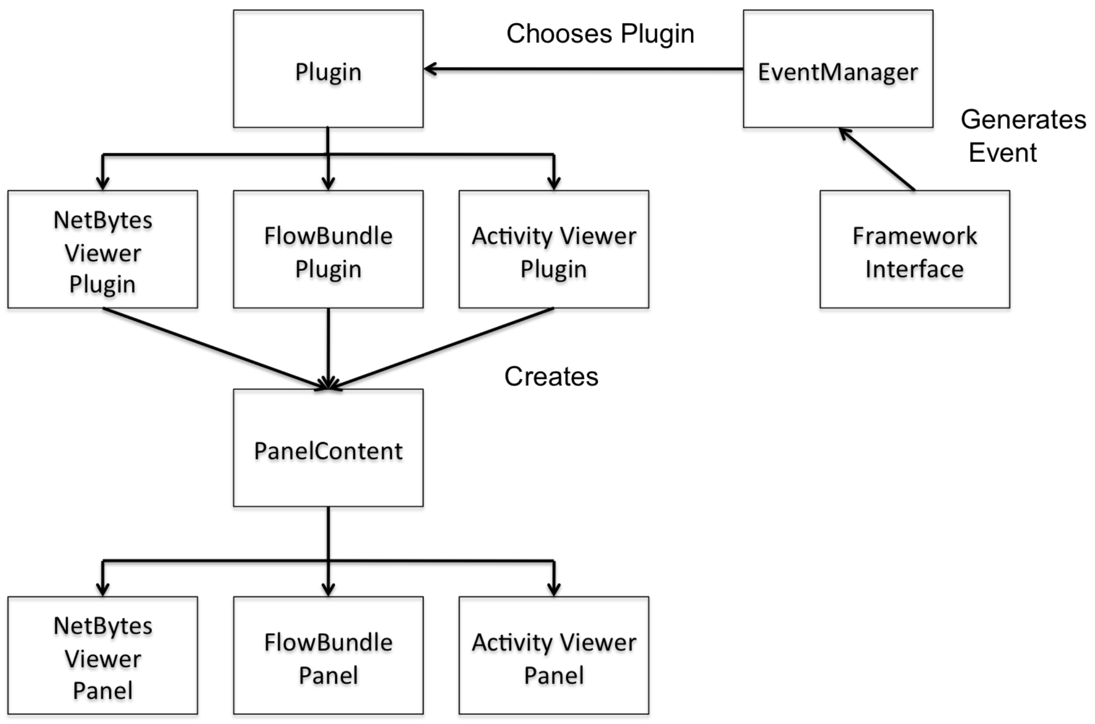


Figure 3.2: High-level framework overview.

by the framework, it is passed to the event manager, which chooses a plugin to handle the event. The plugin then interacts with the user to retrieve information required for creating the visualization. It creates the visualization on a panel (or tabbed pane) and passes it back to the framework for rendering. Visualizations can communicate with one another through the event manager using a plugin and event ID. Data parameters can also be passed from one visualization to another in this manner. For future releases of the framework, there will be an investigation into the creation of a generic data structure to house network attributes such that any visualization can launch any other visualization in the framework even if they are not known to each other at build time (see Chapter 6 for further discussion).

Visualizations can pass data to each other but each visualization handles its own data access. This was done to support data fusion (*i.e.*, allowing the visualization to gather data from several sources).

3.2 FlowBundle

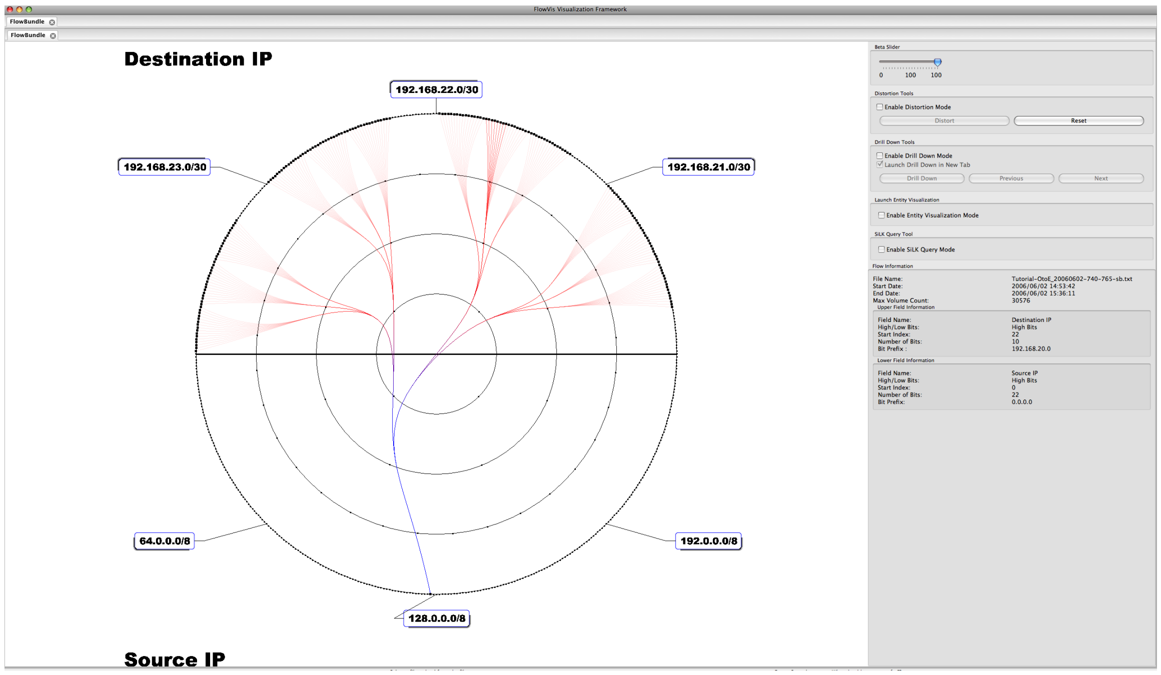
One of the first visualizations built for the FloVis framework, FlowBundle [40], shows connections between any two data attributes (*e.g.*, source IP address and destination IP address, source port and destination port) in a NetFlow record. In Figure 3.3, connections are shown between source and destination IP addresses; however, the user may also want to see source IP address to destination port, to determine which ports each IP address is using as shown in Figure 3.4. Other combinations may be useful (*e.g.*, source IP address to protocol) and are available for viewing with this tool.

FlowBundle was created to address some of the key issues facing current connection-oriented graphs (*i.e.*, parallel coordinates and node-edge graphs). As mentioned in Section 3, these issues include the following:

- Data occlusion caused by connection fanning makes the individual connections indiscernible and data is hidden behind other data points.
- Information overload caused by millions of data points displayed on the screen.
- Lack of labeling because data points are too small and close together.

FlowBundle tries to mitigate these issues by using edge bundling [17] and node aggregation. Edge bundling groups connection lines that are close together (based on data point ordering) and causes the connections to follow predefined paths. These paths show the general flow of traffic between various entities around the circle and help to reduce occlusion under concentrated data loads. In FlowBundle, these paths are determined by a set of control points that are strategically placed around the inner rings of the diagram as shown in Figure 3.3. The control points act like magnets bending the connections along a desired path so that the lines only fan out near the end points. As a result, the user can see the overall data flow. A more detailed discussion on how connection bundling works in FlowBundle is available in Section 4.3.2.

The purpose of node aggregation is to represent multiple data instances in one visual node to reduce the overall number of data points on the screen. For instance, Figure 3.3 shows a set of circular rings with the outer ring containing 512 points.



☐ Enable Distortion Mode
☐ Enable Drill Down Mode
☑ Launch Drill Down in New Tab
☐ Enable Entity Visualization Mode
☐ Enable SiLK Query Mode

Flow Information

```
File Name:                Tutorial-OtoE_20060602-740-765-sb.txt
Start Date:               2006/06/02 14:53:42
End Date:                 2006/06/02 15:36:11
Max Volume Count:         30576
 Upper Field Information
 Field Name:              Destination IP
 High/Low Bits:           High Bits
 Start Index:             22
 Number of Bits:          10
 Bit Prefix :             192.168.20.0
 Lower Field Information
 Field Name:              Source IP
 High/Low Bits:           High Bits
 Start Index:             0
 Number of Bits:          22
 Bit Prefix:              0.0.0.0
```

Figure 3.3: FlowBundle displays data flows between entities.

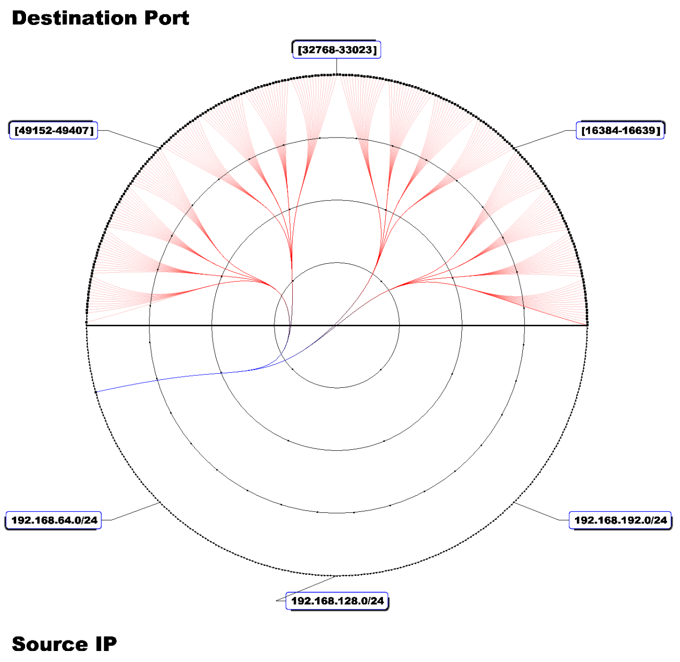


Figure 3.4: FlowBundle displays communication between source IP addresses and destination ports.

The circle is divided into two halves with 256 points along the top and 256 points along the bottom. Each point represents an aggregation of the traffic of the source or destination addresses with the same 8 most significant bits. In other words, hosts whose IP addresses have the same 8 most significant bits are represented by a single node. All connections involving those hosts will end at the same node. This reduces the number of connections drawn on the screen at once thereby reducing occlusion and enabling the use of labeling. As shown in Figure 3.3, a set of strategic points is labeled around the circle so that the visualization is not cluttered. IP addresses are numerically sorted around the circle so that the user can infer the pattern; however, the user can right click on any data point on the circle and a corresponding label will appear. Labels are added along an invisible oval shape around the diagram. As more labels are added, a collision detection algorithm ensures that labels do not overlap. When new labels are added that may overlap, existing labels are pushed along the oval to accommodate them. Furthermore, labels can be removed by right clicking on the data point a second time.

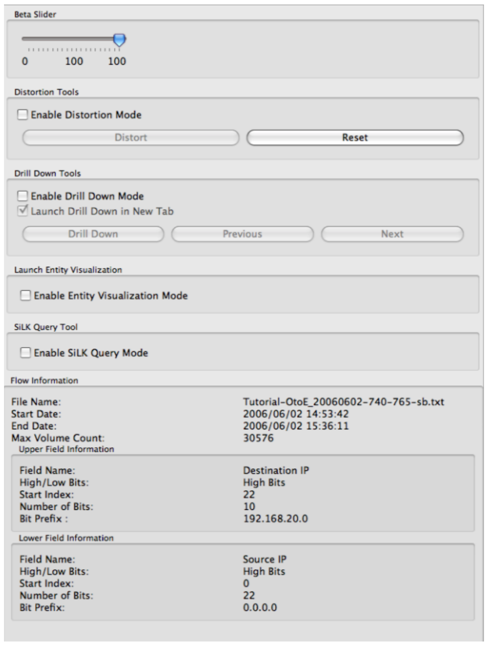
Colour is used to represent flow direction. Flows travel from blue to red. This was done in case the user chooses to show bidirectional flows on a single graph (see Section 3.2.2 for more details). Transparency is used to indicate traffic volume. The more opaque the line, the higher the percentage of traffic that is flowing along that connection. While reading the data, the visualization finds the connection with the maximum traffic volume and stores that value. It then creates a percentage-based quartile using the maximum value. The percentage is divided into ranges: 0 to 25, 25 to 50, 50 to 75 and 75 to 100 percent. Higher ranges are represented with a greater degree of opacity. For instance, if a connection has a volume less than 25 percent of the maximum volume it gets the greatest transparency. This ensures that the more traffic intensive connections are more prominent to the user.

3.2.1 Features

Along the right hand side of FlowBundle is a control panel as shown in Figure 3.5. On the bottom of the panel is an information box which outlines the details of the current dataset loaded. It tells the user the following information:

- The name of the data file currently loaded.

- The time period over which the data was collected.
- The volume value of the connection with the highest traffic level in the data set.
- The details about the data attributes represented in both the top and bottom portions of the circle. This information includes: prefix mask, bit length and SiLK field type.



☐ Enable Distortion Mode

☐ Enable Drill Down Mode
✓ Launch Drill Down in New Tab

☐ Enable Entity Visualization Mode

☐ Enable SiLK Query Mode

File Name:	Tutorial-OtoE_20060602-740-765-sb.txt
Start Date:	2006/06/02 14:53:42
End Date:	2006/06/02 15:36:11
Max Volume Count:	30576

Field Name:	Destination IP
High/Low Bits:	High Bits
Start Index:	22
Number of Bits:	10
Bit Prefix :	192.168.20.0

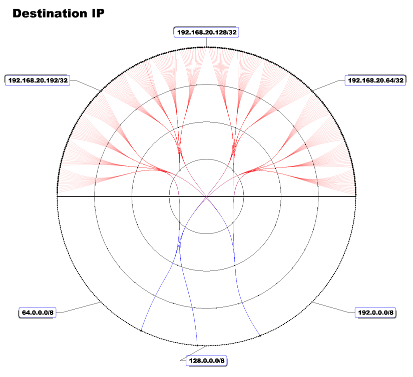
Field Name:	Source IP
High/Low Bits:	High Bits
Start Index:	0
Number of Bits:	22
Bit Prefix:	0.0.0.0

Figure 3.5: The control panel provides information on the current data set visualized as well as interactive buttons for drill down, distortion and other features.

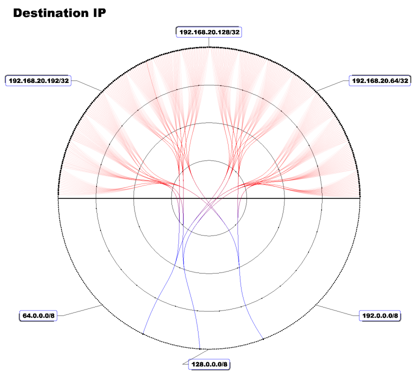
The control panel also contains interactive widgets that allow the user to manipulate the view for a more detailed investigation. One such widget is the beta slider, which controls the tightness (or looseness) of the bundles as shown in Figure 3.6. Bundle loosening straightens the bundles so that connections are viewable on an individual basis. The beta slider controls the bundle tightness to a fine granularity. It ranges from 0 (loose straight lines as shown in Figure 3.6(e)) to 100 (tightest bundles as shown in Figure 3.6(a)). Figure 3.6 shows the progression from the tight bundling to the straight fanning lines that occurs when moving the slider down. Note that this loosening of bundles results in the visual occlusion issue discussed earlier; however, the use of transparency makes high volume connections prevalent in the diagram as shown by the single dark connection in Figure 3.6(e).

FlowBundle also enables the user to activate *drill down mode* through a checkbox on the control panel as seen in Figure 3.5. This mode enables drill down on an aggregate node. The inner rings of the bundle diagram create a radial tree layout for the underlying data source, thus facilitating the ability to drill down within the data. Drill down mode activates a set of transparent green ovals on top of points on the inner rings as shown in Figure 3.7(a). This allows the user to select a specific branch on the diagram (represented with a purple ring in the figure). Clicking on a specific branch drills down into that branch, launching another bundle visualization with that particular branch now expanded to fill the semi-circle. Drilling down involves sliding an 8-bit mask (initially covering the highest 8 bits of the entity's address) down by some number of bits (which is determined by how far along the branch the user clicks in drill down mode). See Section 4.3.2 for more details on how the bit sliding works. Bit sliding reduces the number of IP addresses in a node aggregate and provides more details about individual hosts (see the use case in Section 5.1.2 for an example). Figure 3.7(b) shows a bit slide drill down of two bits. Notice that after drill down, the diagram shows individual IP addresses along the top portion of the circle.

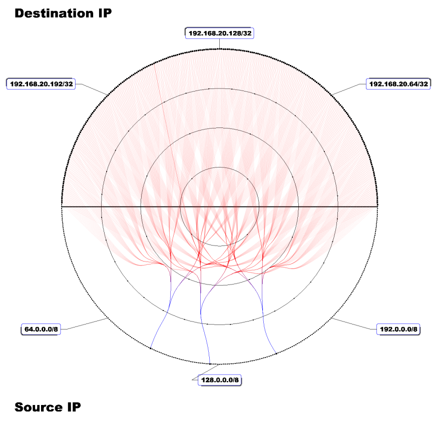
The new drilled down visualizations can be launched in place over the existing diagram, or in a separate tab (or window) to retain a historical context. A technique called hierarchical tabbing is used so that the user can analyze several different connection files while keeping the drilled down visualizations organized. In hierarchical tabbing, there are two levels of tabs: a top level and a lower level as shown in Figure 3.8. The top level tab organizes the visualizations by data file. In other words, each tab at the high level represents a visualization plugin (FlowBundle, NetBytes Viewer, etc.) which has loaded a different set of data. The lower level tabs show the different modifications to the same visualization and the same dataset. For instance, the user may want to investigate all the connections from her internal network to the Internet for June 29th, 2006. This would create a top level tab and a subsequent lower level with the visualization. If the user then wanted to drill down on the visualization, that would open another low level tab to give the user historical context for the drill down. If the user wanted to open the connection data for June 30th, 2006, this would launch



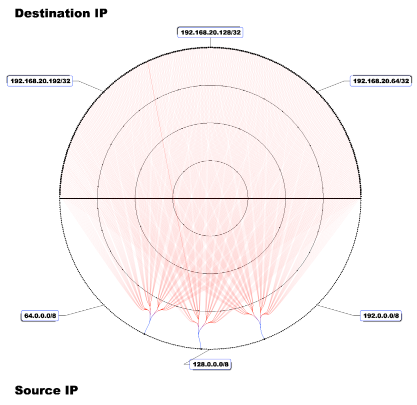
(a) Beta slider at 100.



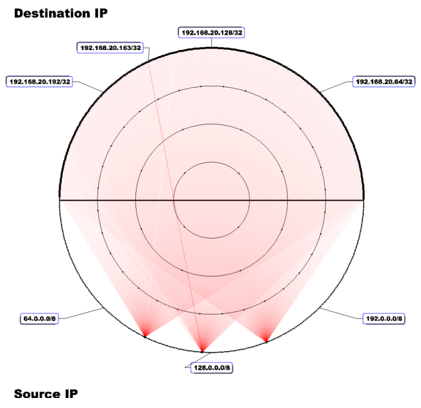
(b) Beta slider at 90.



(c) Beta slider at 50.

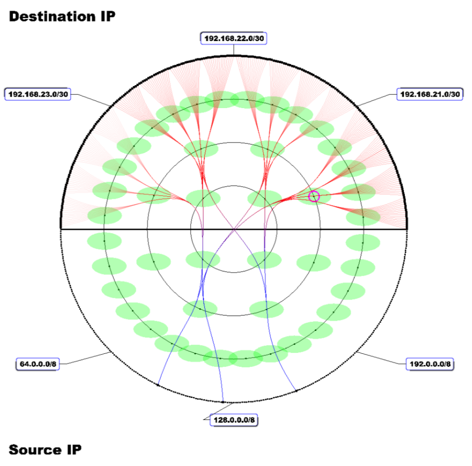


(d) Beta slider at 15.

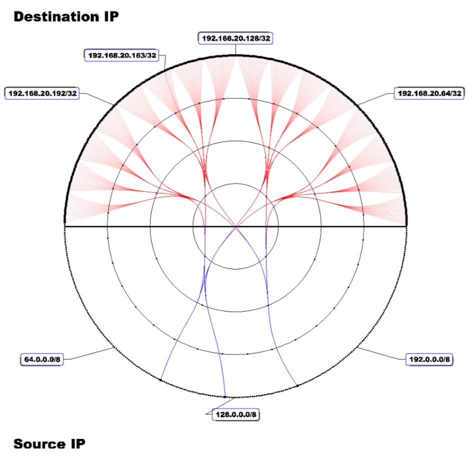


(e) Beta slider at 0.

Figure 3.6: FlowBundle bundle loosening with the beta slider.



(a) FlowBundle in drill down mode. Notice the pink ring on the top right corner of the circle.



(b) Drill down result. FlowBundle now shows individual hosts after the bit window is shifted by 2-bits.

Figure 3.7: FlowBundle in drill down mode.

another top level tab for the new dataset. In Figure 3.8, there is a tab for FlowBundle and the Activity Viewer. Currently, the user is clicked on the FlowBundle tab and has two drill down tabs associated with the dataset.

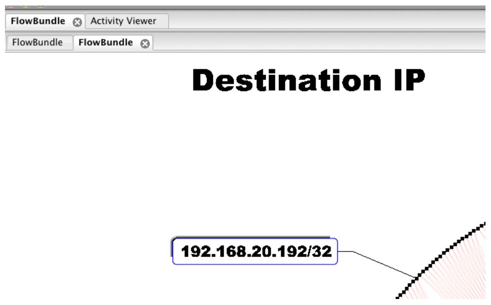
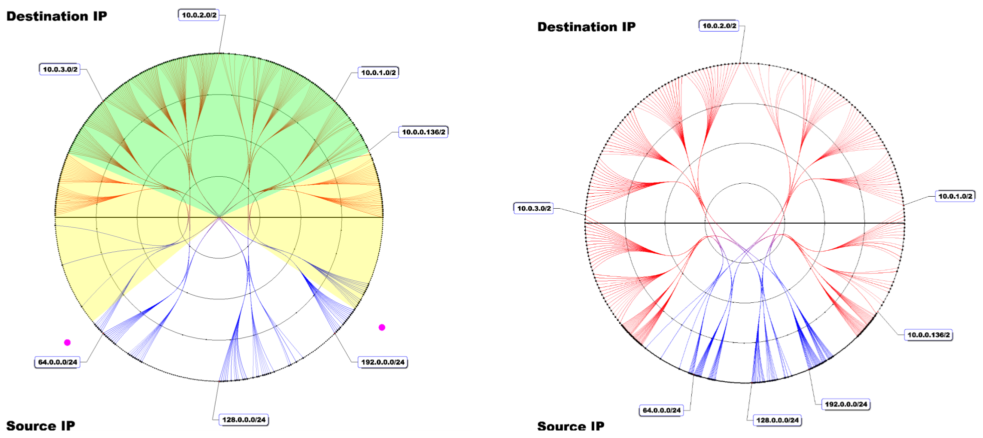


Figure 3.8: Hierarchical tabbing example. The top level tabs group by visualization while the lower level tab show the same visualization but modified by drill down.

Linear distortion mode is another feature which is enabled by a checkbox on the FlowBundle control panel. The feature provides the ability to linearly distort points on the circle as shown in Figure 3.9. Distortion [25] is a well known presentation technique that allows users to expand an area and focus on key points of a visualization that may show interesting behavior, while pushing aside points that are not interesting. After enabling distortion mode, the user must decide which region he wants to give spatial focus. He does this by clicking on the mouse and dragging it over the desired region. This creates a green highlight as shown in Figure 3.9(a). Next, the user decides how much additional space to allocate for the region. He does this by clicking on each of the two purple dots which reside at either end of the green region. The user drags the dots to create the expanded region which becomes the green and yellow region in Figure 3.9(a). After clicking the distortion button on the right hand panel, the visualization changes to Figure 3.9(b). Emphasis is placed on the upper data points; they are given more space along the circle for further analysis while the lower data points are compressed into the bottom eighth of the circle.



(a) The first step in linear distortion is to highlight the region of the circle to expand spatially (green area). Then decide how much space to give the points (yellow + green section).

(b) Linear distortion puts focus on key points on the circle.

Figure 3.9: Linear distortion mode enables the user to focus on key points.

Finally, FlowBundle offers an *entity visualization mode* which is also enabled by a checkbox on the right hand panel. In this mode, the user can double click on any of the IP addresses on the outer part of the circle and launch the NetBytes Viewer based on the selected host or subnet. After double clicking on the host, a configuration dialog (described in the next section) appears enabling the user to change the configuration settings for the NetBytes Viewer.

FlowBundle typically takes a daily or hourly connection bag as its input. Connection bags contain information about the host-to-host connections occurring during a particular day or hour and can contain byte, flow or packet volume information for each connection. More detailed information on the FlowBundle implementation and SiLK data processing is available in Section 4.4.

3.2.2 Bidirectional Flows

As mentioned earlier, FlowBundle can show bidirectional connections; however, the overlapping bundles make it difficult to discern direction, even when using the colour change (blue to red) to signify direction. One approach to solving this problem is to create two FlowBundle diagrams side-by-side — one showing traffic in one direction and the other showing traffic in the other. This helps to discern direction, but it can be difficult to match corresponding data points on both circles. As a result, a bidirectional bundle visualization was constructed as shown in Figure 3.10.

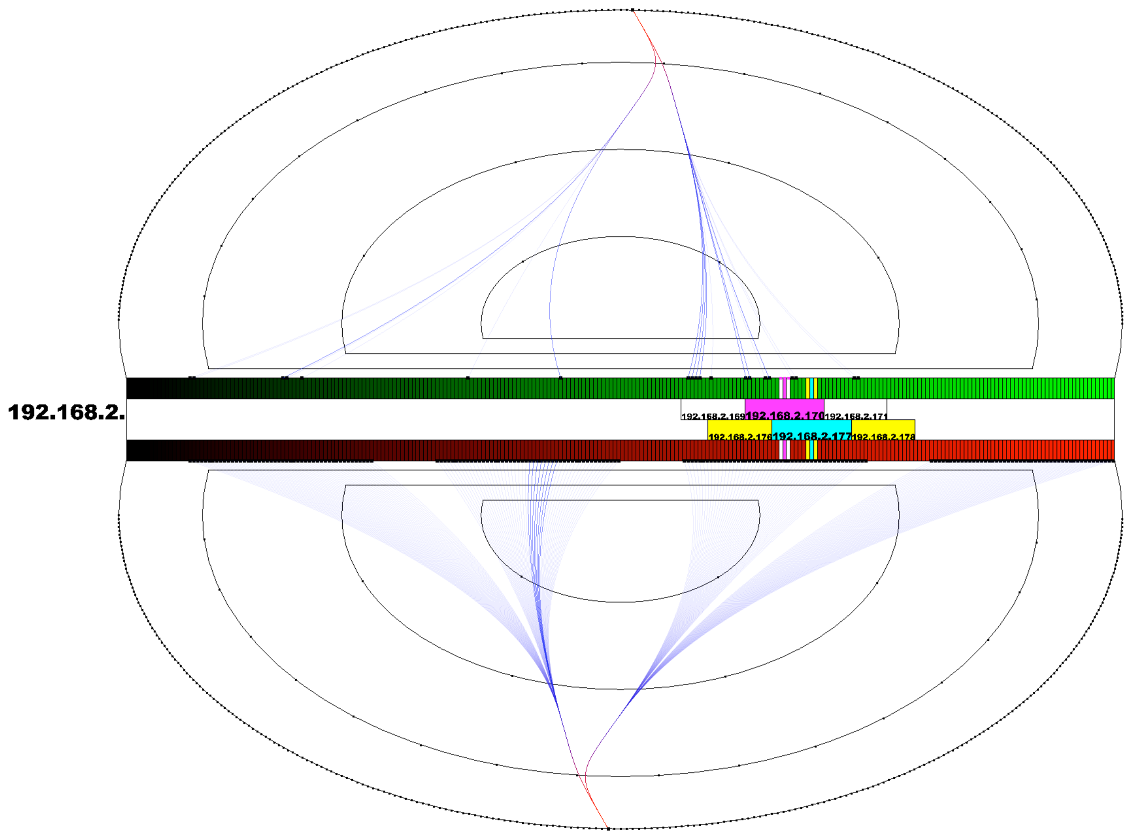


Figure 3.10: Bidirectional bundle diagram of an external scanner. External hosts are on the outer circle; internal hosts along the centre. Bottom is external to internal flow. Top is internal to external flow. Notice the two sets of labels which slide along the middle bars to identify data points.

The top and bottom portions of the diagram have 256 points each and represent the external network. Each point on the top portion has a mirrored opposite point on the bottom portion. In the middle, there is a green strip and a red strip each divided into 256 blocks. Each green block with the corresponding red block below it represents an internal host. Connections from the internal host to the external host are made from the top of the green blocks to the top portion of the circle. Connections from the external hosts to the internal hosts are made from the bottom circle to the red connection. A colour gradient is used along the red and green strips in order to help delineate between the individual blocks.

Labeling the internal blocks in this scenario is much more difficult since the data points are much closer together. As a result, a sliding ruler labeling paradigm was adopted. When a user places her mouse over an individual block on the green band, that block changes colour (as well as the block before and after) and three corresponding labels are shown in the space below (shown in Figure 3.10). As the user moves the mouse across the band, the labels follow. A second set of corresponding labels follow the red band. This gives the user a total of six possible labels that she can interactively maneuver during analysis. Other interactive features include the ability to loosen the bundles as in the unidirectional FlowBundle diagram. Drill down capabilities are desired for this visualization and are allotted for future work.

3.3 NetBytes Viewer

NetBytes Viewer [38] is intended to fill another gap left by current security visualizations. Current visualizations do not offer much in the way of drill down to the host level. VisFlowConnect [42], Portall [11] and Flamingo [31] do not have any drill down capabilities while VISUAL [3] has a limited text-based drill down. NetBytes Viewer is meant to compliment FlowBundle by providing a more detailed drill down analysis of individual entities (hosts). While FlowBundle focuses on the interactions between entities (host or subnet) on a network, NetBytes focuses on the traffic volume entering and leaving a single entity in more detail. It is inspired by a static visualization described by McHugh et al. [29].

As shown in Figure 3.11, NetBytes Viewer is a 3D impulse graph with three axes: Time versus Port/Protocol versus Volume. This gives a *picket fence* effect where each colour in the diagram represents the hourly traffic volume for a particular port or protocol over a specific time period (*e.g.*, weeks or months). NetBytes Viewer cycles through a set of ten different colours and then repeats. This helps to more easily distinguish between activity on ports (or protocols) that are close together. Volumes are shown as impulses (or spikes) using a log scale so that small volumes (order of 1 to 100) are as prevalent in the visualization as large volumes (1,000,000 or more). Adding a time dimension to the diagram provides historical context. It allows the user to see what traffic occurred on which port/protocol over time so that she can see changes in volume or port behaviour without having to rely on the use of animation. Animation can cause *change blindness* [35], which suppresses the user's ability to see small changes in the data. It also relies on the user's short term memory to remember patterns while with NetBytes Viewer, the patterns are immediately presented to the user.

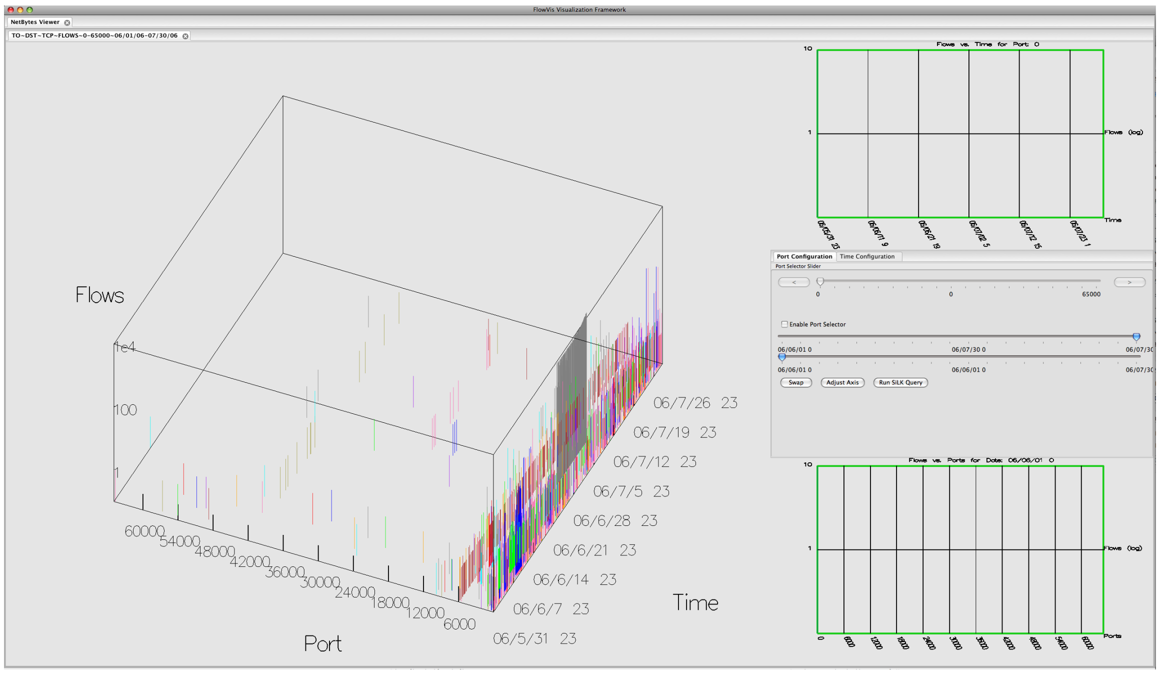
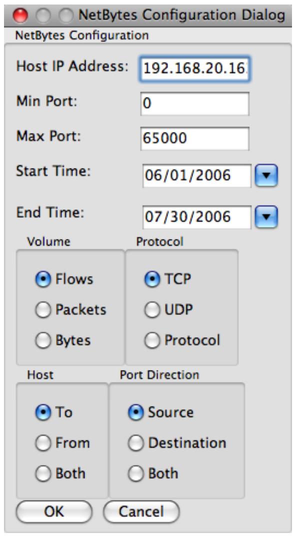


Figure 3.11: NetBytes Viewer is a 3D impulse graph: Time vs. port/protocol vs. Volume (flows, bytes or packets).

NetBytes Viewer is launchable from the main menu of the FloVis framework or by clicking on a host in FlowBundle. This launches a configuration dialog as shown in Figure 3.12 which enables the user to filter the data and set port/protocol as well as date ranges. He can analyze volume going to or coming from a host along with selecting source or destination ports. The user can also filter by TCP or UDP data. To facilitate faster rendering, NetBytes Viewer is built on top of a MySQL database that has been preloaded with port and protocol bags for individual hosts or subnetworks.



NetBytes Configuration

Host IP Address: 192.168.20.16

Min Port: 0

Max Port: 65000

Start Time: 06/01/2006

End Time: 07/30/2006

Volume: [✓ Flows] Packets Bytes

Protocol: [✓ TCP] UDP Protocol

Host: [✓ To] From Both

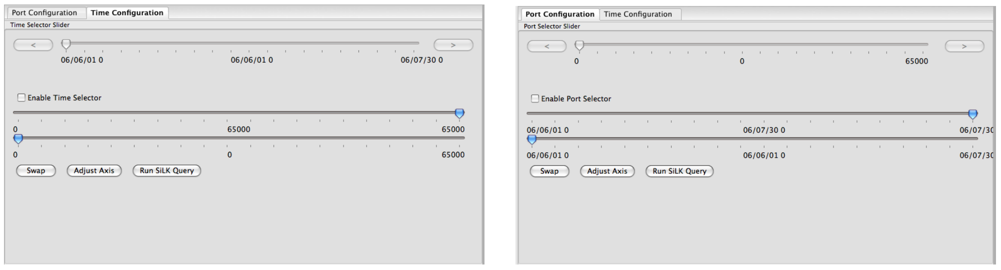
Port Direction: [✓ Source] Destination Both

Figure 3.12: Configuring the NetBytes Viewer from a dialog.

3.3.1 Features

Three dimensional graphs have some issues that must be addressed. First, they suffer from the aforementioned data occlusion problem, when data is rendered behind other data in a 3D space. Loss of depth perception and loss of head parallax [41] are also issues when 3D objects are rendered on 2D screens. To deal with these issues, NetBytes Viewer has added some interactive features. Users are able to rotate the 3D graph in all directions using the mouse. This allows the user to view the data at all angles which can be used to overcome most data occlusion.

The rest of the interactive features are available on a righthand control panel which appears in the middle of two 2D visualization screens as shown in Figure 3.11. The panel is tabbed to keep interactive features for the time (see Figure 3.13(a)) and port/protocol (see Figure 3.13(b)) dimensions together while maximizing screen space for the visualizations.



(a) The time configuration tabbed panel has interactive controls for selection mode, axis adjustment and swapping.

(b) The protocol configuration tabbed panel has interactive controls for selection mode, axis adjustment and swapping.

Figure 3.13: Configuration tabs support NetBytes Viewer interactive features.

One such interactive feature is called *selection mode* and is enabled by clicking on the *Port Selector* or *Time Selector* checkboxes in the control panel. The selection mode feature enables a user to highlight any axis (Time versus Volume or Port/Protocol versus Volume) with a semi-transparent (green for port selection; blue for time selection) plane that slides across the chosen axis. The highlighted axes are then visible in smaller 2D graphs on the right side of the visualization as shown in Figure 3.14. This has the effect of slicing the 3D graph into smaller 2D graphs which can be easier to analyze.

In Figure 3.14, a large block of traffic on port 3306 is highlighted with a transparent green highlight, and this data is shown in the image on the upper right hand corner of the application. This shows the activity on port 3306 over a time period from June 1st, 2006 to July 31st, 2006. A similar highlight (blue) is shown on the volume-port/protocol axis for July 3rd, 2006 at 7 pm. The corresponding image is available in the lower right hand corner of the application and shows the ports active for that time slice. A slider (one for each selector) is used to move the highlighters across the axis, and there are two "snap" buttons (up and down) that snap the highlighter to the next port (or time period) with data (shown at the top of Figures 3.13(a) and 3.13(b)), thus allowing the user to move the highlight in the vicinity of the data he wants to analyze and then use the buttons to hit the data point precisely.

The 2D visualizations in Figure 3.14 (right) are too small for detailed analysis.

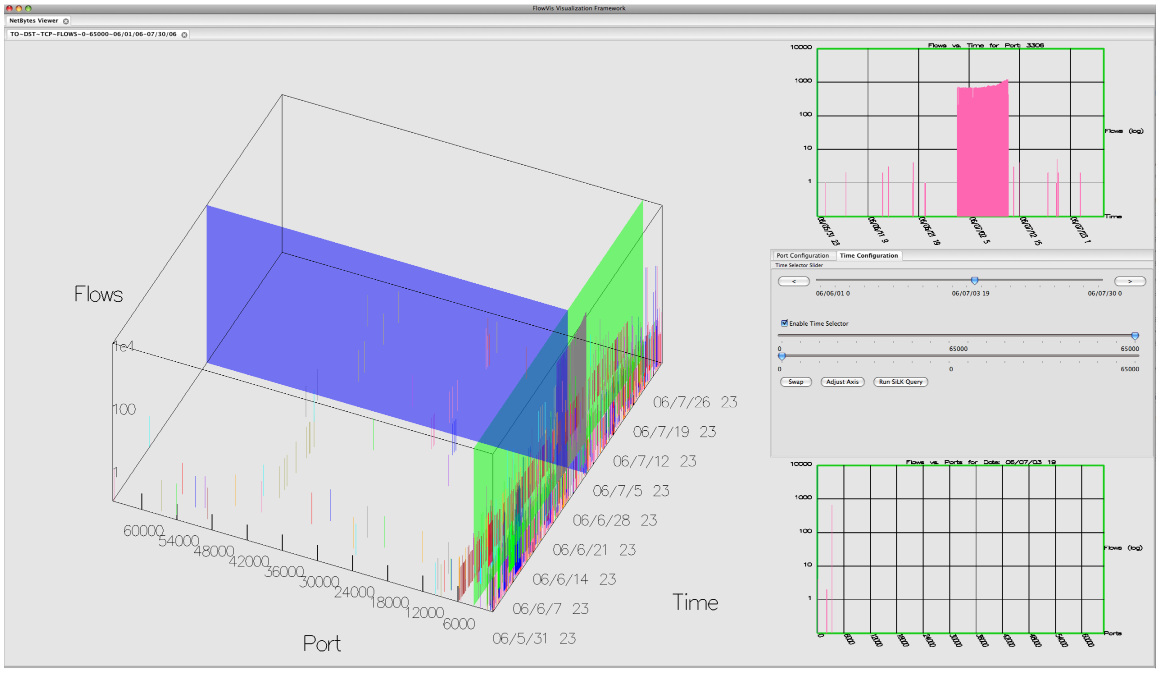


Figure 3.14: NetBytes Viewer selection mode.

As a result, a swapping feature was implemented to allow the user to swap the top or bottom graphs with the main graph as shown in Figure 3.15. All of the interactive features are available after a swap has occurred. The swap feature is activated by pressing the *Swap* button on either the port or time configuration panels.

NetBytes Viewer provides the option to zoom in on a smaller portion of the graph by selecting upper and lower boundaries on both the time and ports/protocol axes using the boundary sliders on the port (time) configuration panel. One slider controls the upper boundary while the other controls the lower boundary. Figure 3.15 shows the sliders (one for the upper boundary and one for the lower boundary) used to highlight a section of the graph, and can be performed on both 2D graphs simultaneously. Once the boundaries are set, the user can relaunch the visualization in another tab with the new, smaller axes using the *Adjust Axes* button on the configuration panel. The adjusted port axis for the highlight from Figure 3.15 is shown in Figure 3.16. This action is intended to allow a user to focus on an area of the graph. Tabs are used so that the user can go back through a hierarchy of zoomed images and set the boundaries on another portion of the graph. Hierarchical tabbing also allows data for multiple hosts to be loaded into the tool simultaneously.

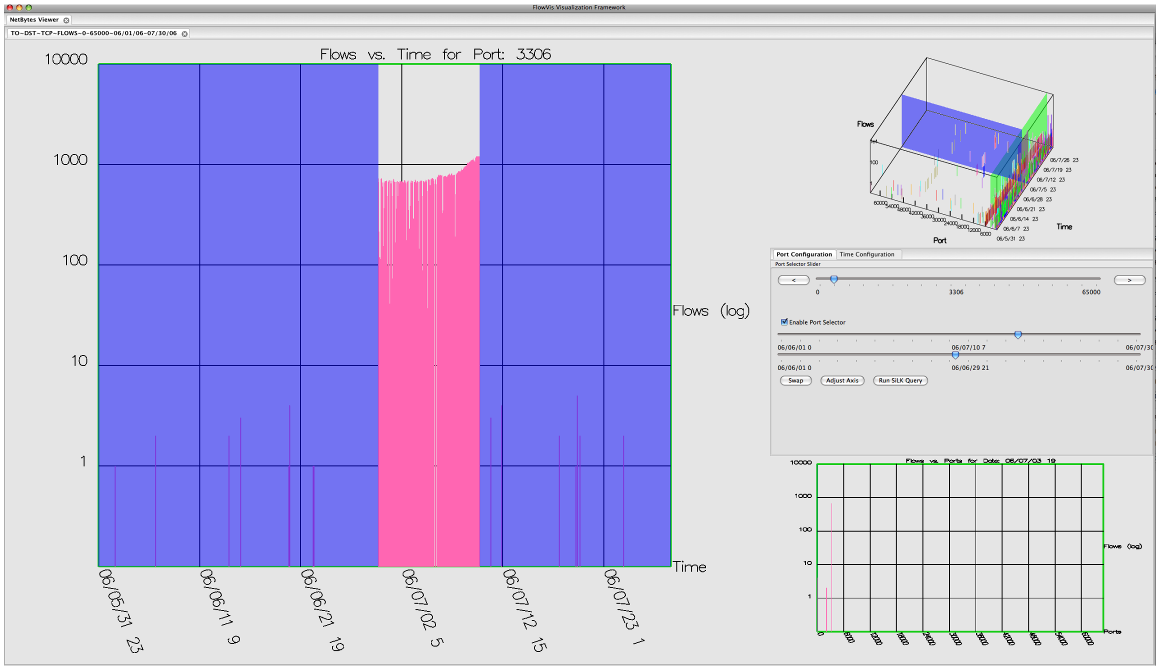


Figure 3.15: Highlighting the upper and lower boundaries of the time axis allows the user to zoom in on the data.

In summary, NetBytes Viewer provides a historical overview of the traffic to and from an entity. For example, if the user is looking at the graph of an email server, the application will show traffic on main ports used in email services (*i.e.*, ports 25 and 110) as well as on DNS ports for address lookup. Traffic observed on atypical ports, or traffic patterns that change over time, could be an indication of compromise or malicious behaviour.

3.4 SiLK Query Tool

Goodall mentions in his user study [14] that it is extremely important to give the user access to the raw underlying data for any visualization. This was confirmed independently during our discussions with analysts. As a result, another plugin was created for the FloVis framework called the SiLK Query Tool. The tool, shown in Figure 3.17, provides an interface to the SiLK tool set. It displays SiLK records in a tabular form using a grid control for simple scrolling by the user. The text box at the top of the display allows users to create their own SiLK queries and view the results in the interface.

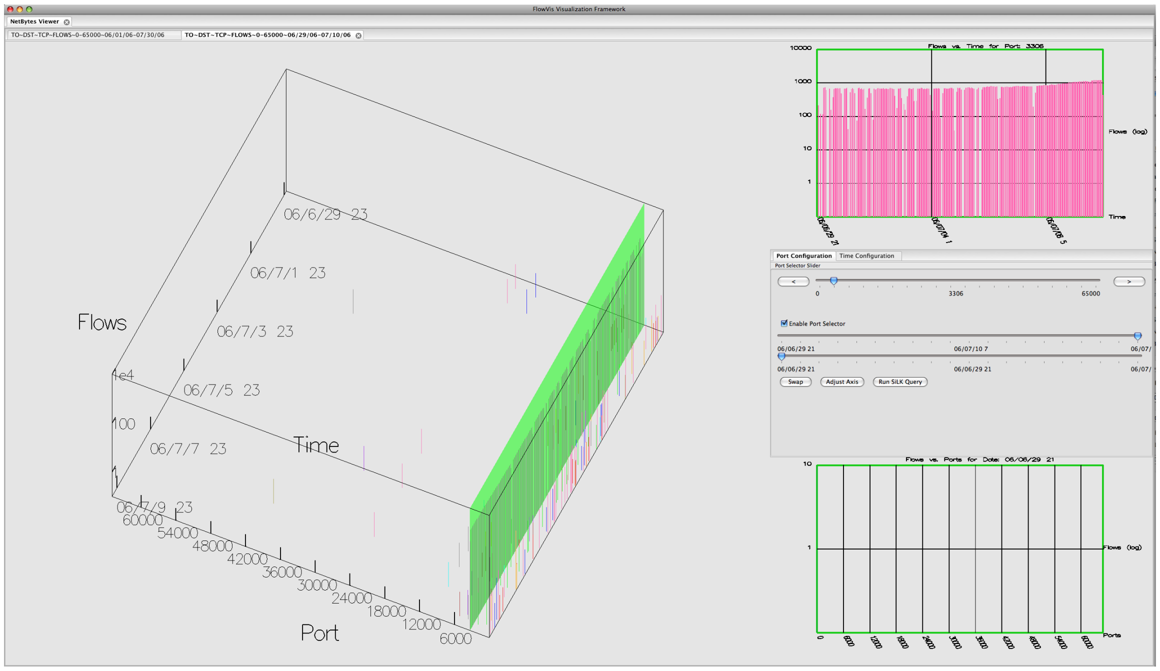
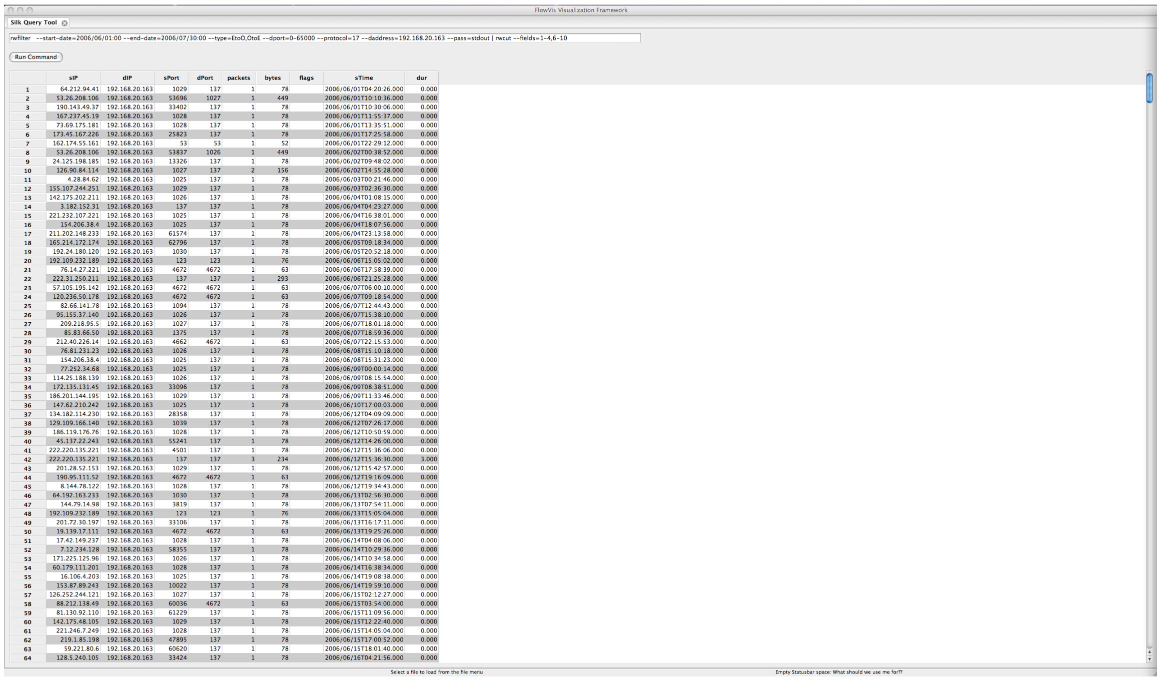

FlowVis Visualization Framework
NetBytes Viewer
TO-DST-TCP-FLOWS-0-65000-06/01/06-07/30/06
TO-DST-TCP-FLOWS-0-65000-06/29/06-07/10/06
Flows
Time
Port
Flows vs. Time for Port: 3306
Flows (log)
Port Configuration
Time Configuration
Port Selector Slider
Enable Port Selector
Swap
Adjust Axis
Run SiLK Query
Flows vs. Ports for Date: 06/06/29 21
Ports

Figure 3.16: The adjusted time axis spreads the traffic on port 3306 over a larger space.



Silk Query Tool

```
rwfilter   --start-date=2006/06/01:00 --end-date=2006/07/30:00 --type=EtoO,OtoE --dport=0-65000 --protocol=17 --daddress=192.168.20.163 --pass=stdout | rwcut --fields=1-4,6-10
```

Run Command

	sIP	dIP	sPort	dPort	packets	bytes	flags	sTime	dur
1	64.212.94.41	192.168.20.163	1029	137	1	78		2006/06/01T04:20:26.000	0.000
2	53.26.208.106	192.168.20.163	53696	1027	1	449		2006/06/01T10:10:36.000	0.000
3	190.143.49.37	192.168.20.163	33402	137	1	78		2006/06/01T10:30:06.000	0.000
4	167.237.45.19	192.168.20.163	1028	137	1	78		2006/06/01T11:55:37.000	0.000
5	73.69.175.181	192.168.20.163	1028	137	1	78		2006/06/01T13:35:51.000	0.000
6	173.45.167.226	192.168.20.163	25823	137	1	78		2006/06/01T17:25:58.000	0.000
7	162.174.55.161	192.168.20.163	53	53	1	52		2006/06/01T22:29:12.000	0.000
8	53.26.208.106	192.168.20.163	53837	1026	1	449		2006/06/02T00:38:52.000	0.000
9	24.125.198.185	192.168.20.163	13326	137	1	78		2006/06/02T09:48:02.000	0.000
10	126.90.84.114	192.168.20.163	1027	137	2	156		2006/06/02T14:55:28.000	0.000
11	4.28.84.62	192.168.20.163	1025	137	1	78		2006/06/03T00:21:46.000	0.000
12	155.107.244.251	192.168.20.163	1029	137	1	78		2006/06/03T02:36:30.000	0.000
13	142.175.202.211	192.168.20.163	1026	137	1	78		2006/06/04T01:08:15.000	0.000
14	3.182.152.31	192.168.20.163	137	137	1	78		2006/06/04T04:23:27.000	0.000
15	221.232.107.221	192.168.20.163	1025	137	1	78		2006/06/04T16:38:01.000	0.000
16	154.206.38.4	192.168.20.163	1025	137	1	78		2006/06/04T18:07:56.000	0.000
17	211.202.148.233	192.168.20.163	61574	137	1	78		2006/06/04T23:13:58.000	0.000
18	165.214.172.174	192.168.20.163	62796	137	1	78		2006/06/05T09:18:34.000	0.000
19	192.24.180.120	192.168.20.163	1030	137	1	78		2006/06/05T20:52:18.000	0.000
20	192.109.232.189	192.168.20.163	123	123	1	76		2006/06/06T15:05:02.000	0.000
21	76.14.27.221	192.168.20.163	4672	4672	1	63		2006/06/06T17:58:39.000	0.000
22	222.31.250.211	192.168.20.163	137	137	1	293		2006/06/06T21:25:28.000	0.000
23	57.105.195.142	192.168.20.163	4672	4672	1	63		2006/06/07T06:00:10.000	0.000
24	120.236.50.178	192.168.20.163	4672	4672	1	63		2006/06/07T09:18:54.000	0.000
25	82.66.141.78	192.168.20.163	1094	137	1	78		2006/06/07T12:44:43.000	0.000
26	95.155.37.140	192.168.20.163	1026	137	1	78		2006/06/07T15:38:10.000	0.000
27	209.218.95.5	192.168.20.163	1027	137	1	78		2006/06/07T18:01:18.000	0.000
28	85.83.66.50	192.168.20.163	1375	137	1	78		2006/06/07T18:59:36.000	0.000
29	212.40.226.14	192.168.20.163	4662	4672	1	63		2006/06/07T22:15:53.000	0.000
30	76.81.231.23	192.168.20.163	1026	137	1	78		2006/06/08T15:10:18.000	0.000
31	154.206.38.4	192.168.20.163	1025	137	1	78		2006/06/08T15:31:23.000	0.000
32	77.252.34.68	192.168.20.163	1025	137	1	78		2006/06/09T00:00:14.000	0.000
33	114.25.188.139	192.168.20.163	1026	137	1	78		2006/06/09T08:15:54.000	0.000
34	172.135.131.45	192.168.20.163	33096	137	1	78		2006/06/09T08:38:51.000	0.000
35	186.201.144.195	192.168.20.163	1029	137	1	78		2006/06/09T11:33:46.000	0.000
36	147.62.210.242	192.168.20.163	1025	137	1	78		2006/06/10T17:00:03.000	0.000
37	134.182.114.230	192.168.20.163	28358	137	1	78		2006/06/12T04:09:09.000	0.000
38	129.109.166.140	192.168.20.163	1039	137	1	78		2006/06/12T07:26:17.000	0.000
39	186.119.176.76	192.168.20.163	1028	137	1	78		2006/06/12T10:50:59.000	0.000
40	45.137.22.243	192.168.20.163	55241	137	1	78		2006/06/12T14:26:00.000	0.000
41	222.220.135.221	192.168.20.163	4501	137	1	78		2006/06/12T15:36:06.000	0.000
42	222.220.135.221	192.168.20.163	137	137	3	234		2006/06/12T15:36:30.000	3.000
43	201.28.52.153	192.168.20.163	1029	137	1	78		2006/06/12T15:42:57.000	0.000
44	190.95.111.52	192.168.20.163	4672	4672	1	63		2006/06/12T19:16:09.000	0.000
45	8.144.78.122	192.168.20.163	1028	137	1	78		2006/06/12T19:34:43.000	0.000
46	64.192.163.233	192.168.20.163	1030	137	1	78		2006/06/13T02:56:30.000	0.000
47	144.79.14.98	192.168.20.163	3819	137	1	78		2006/06/13T07:54:11.000	0.000
48	192.109.232.189	192.168.20.163	123	123	1	76		2006/06/13T15:05:04.000	0.000
49	201.72.30.197	192.168.20.163	33106	137	1	78		2006/06/13T16:17:11.000	0.000
50	19.139.17.111	192.168.20.163	4672	4672	1	63		2006/06/13T19:25:26.000	0.000
51	17.42.149.237	192.168.20.163	1028	137	1	78		2006/06/14T04:08:06.000	0.000
52	7.12.234.128	192.168.20.163	58355	137	1	78		2006/06/14T10:29:36.000	0.000
53	171.225.125.96	192.168.20.163	1026	137	1	78		2006/06/14T10:34:58.000	0.000
54	60.179.111.201	192.168.20.163	1028	137	1	78		2006/06/14T16:38:34.000	0.000
55	16.106.4.203	192.168.20.163	1025	137	1	78		2006/06/14T19:08:38.000	0.000
56	153.87.89.243	192.168.20.163	10022	137	1	78		2006/06/14T19:59:10.000	0.000
57	126.252.244.121	192.168.20.163	1027	137	1	78		2006/06/15T02:12:27.000	0.000
58	88.212.138.49	192.168.20.163	60036	4672	1	63		2006/06/15T03:54:00.000	0.000
59	81.130.92.110	192.168.20.163	61229	137	1	78		2006/06/15T11:09:56.000	0.000
60	142.175.48.105	192.168.20.163	1029	137	1	78		2006/06/15T12:22:40.000	0.000
61	221.246.7.249	192.168.20.163	1028	137	1	78		2006/06/15T14:05:04.000	0.000
62	219.1.85.198	192.168.20.163	47895	137	1	78		2006/06/15T17:00:52.000	0.000
63	59.221.80.6	192.168.20.163	60620	137	1	78		2006/06/15T18:01:40.000	0.000
64	128.5.240.105	192.168.20.163	33424	137	1	78		2006/06/16T04:21:56.000	0.000

Select a file to load from the file menu

Empty Statusbar space: What should we use me for??

Figure 3.17: The SiLK Query Tool is a front end to the SiLK tool set.

The SiLK Query Tool can be launched independently from the FloVis Framework menu or from the FlowBundle and NetBytes Viewer visualizations. In the latter case, a SiLK query is automatically created with the parameters used to generate the visualizations. The user is able to change the automatic query to find different results. Since SiLK queries can take significant time to process, the query tool is run on a separate thread so that the user can still use the application while the query runs in the background.

Chapter 4

Implementation

The FloVis architecture is realized as an interacting set of applications that operate from a common framework. This allows the user to pivot between views of the data. In this chapter, the implementation and underlying data structures that support the framework are described.

4.1 Technical Specification - Framework

The FloVis framework was designed to be an open-source, platform-independent architecture for the integration of security visualizations. It was built using a platform independent windowing system called Wx-Widgets[1]. Wx-Widgets was chosen because it supports Windows, Unix and Macintosh operating systems such that most user needs are satisfied. The windowing system was also chosen because it was written in C++ and is compiled natively for each operating system. This gives the application a memory and speed advantage over applications written in JAVA. Memory issues are of concern because of the large datasets utilized.

Although it is mainly a windowing system, FloVis offers other important features including APIs for the following:

- Configuration Management: manages access to a configuration file so that each visualization has access to custom configuration settings.
- Logging: central and consistent logging mechanism.
- Event Management: managing events between visualizations.
- Font Management: maintains all the fonts used in visualizations.

As mentioned previously, visualizations are built as plugins to the framework. In this way, they follow a plugin interface as shown in Figure 4.1.

[1] `http://www.wxwidgets.org/`

```
class FVPlugin {
 public:
   FVPlugin(string name, string id){ m_name = name; m_id = id;}
   virtual ~FVPlugin(){}
   virtual wxWindow* CreateTab(wxWindow* parent, int frameId,
   int eventId, FVEventArgs* eventArgs) = 0;
   virtual void ModifyTab(FVBasePanel* panel, int parentFrameId,
   int eventId, FVEventArgs* eventArgs) = 0;
   virtual wxWindow* CreateTabFromMenu(wxWindow* parent,
   int parentFrameId) = 0;
   string GetName(){return m_name;}
   string GetId(){return m_id;}
 protected:
   string m_name;
   string m_id;
};
```

Figure 4.1: Framework abstract plugin class interface.

Each visualization must create an object which subclasses the *FVPlugin* object. The *CreateTabFromMenu* method is called by the framework when the user chooses the plugin from the menu. At this point, the plugin gathers the necessary information from the user required to render the visualization. It builds an object subclassing wxWindow (typically a wxPanel or TabbedPanel) and passes it to the framework which handles displaying it to the user. *CreateTab* and *ModifyTab* are used when trying to launch a new visualization from an existing one or modifying an existing visualization respectively. Both allow for data parameters to be passed to the user by subclassing the *FVEventArgs* with the plugin's own set of parameters. This allows the system to be flexible in what types of visualizations and data types it can manage.

4.2 Technical Specification - Visualizations

The visualization pipeline [5] (shown in Figure 4.2) is used to describe the step-by-step process of taking raw data and transforming it into its visual representation. Raw data is manipulated (and filtered) into a data table and a set of meta-data is added to describe its structure. The data is then mapped into visual structures that are used to render the geometric shapes seen in the view. Finally, the data is rendered to the user under the current viewing conditions. Throughout the process, the user can interact with the application to filter the data and manipulate the visualizations.

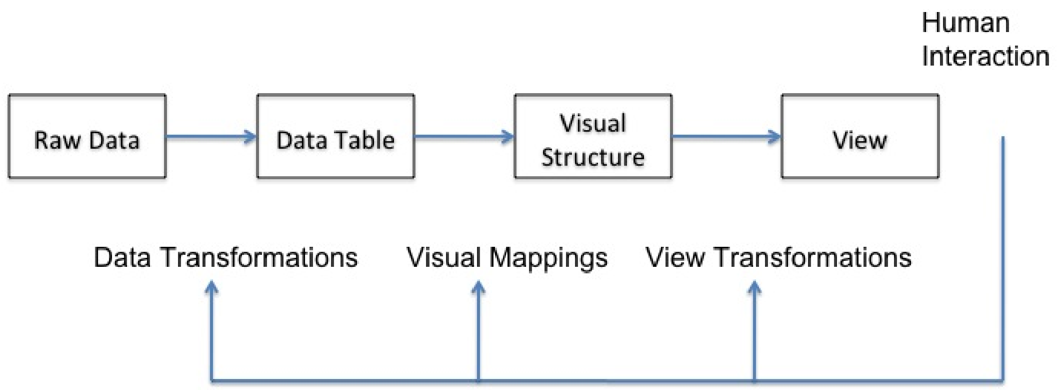


Figure 4.2: The visualization pipeline [5].

In the following sections, the technical aspects of FlowBundle and the NetBytes Viewer are discussed. Each section is organized using headings from the visualization pipeline shown in Figure 4.2. The underlying raw data for all the visualizations discussed here comes from bags generated through the SiLK analysis toolkit described in Section 2.1.

4.3 FlowBundle

FlowBundle is a visualization designed to show the interactions (*i.e.*, connections) between nodes (*i.e.*, hosts or subnets) on a network. It incorporates edge bundling [17] and node aggregation in order to limit visual clutter. FlowBundle uses a circular layout in which the top portion of the circle represents the external hosts and the lower portion represents the internal hosts. Inside the circle is a radial tree structure that serves as a path to direct connection lines between sources and destinations. FlowBundle also incorporates a set of interactive features, including drill down and linear distortion, to allow the the user to examine data in detail. It is also integrated with NetBytes Viewer to allow analysis of volume data in more detail.

This section outlines some of the key implementation details for FlowBundle. In particular, data extraction (from the SiLK tool suite), manipulation, and rendering are discussed.

4.3.1 Data Table

The input to the FlowBundle application is a bag that is indexed by 32-bit values taken from two scalar fields of the flow records[2]. The `rwconnectbag` tool, which was added to the SiLK tool set, creates bags that are indexed by (up to) 32 bits which are taken from arbitrary bit positions in any two scalar fields in the flow records. In other words, portions of two scalar fields (*e.g.*, source address and destination port) from a flow record are concatenated together to form one 32-bit value. This value is used as an index for a SiLK bag to measure the level of activity between the entities that contribute to the two fields. The most popular two scalar fields are the source and destination IP addresses. In this case, a portion of the 32-bit value would go to the source address and the rest to the destination. As an example, the configuration could be used to represent connections from subnet to subnet (*e.g.*, /16 to /16), subnet to host within a given subnet (*e.g.*, /24 prefix to hosts in a given /24), or host to host within two prescribed subnets (*e.g.*, hosts within a given /8 to hosts within a /24). For each 32-bit value in the bag, the traffic activity between the entities represented by the two fields in the 32-bit value is accumulated over a time period (typically an hour or a day, but arbitrary time intervals can be specified). In addition to flows between addresses, the user can visualize the relationship between any two scalars, such as the destination ports targeted by a set of source IP addresses. All of this is done in an effort to offer the utmost flexibility to the analyst using FlowBundle.

A typical input file to FlowBundle contains a header with the following information:

- The bit ranges for both the high and low order bits of the 32-bit value indices in the file.

- The SiLK field numbers representing the field types (*e.g.*, source address, source port, etc.) of the two scalar fields that are encoded into the 32-bit value.

- The mask value (if any) to be OR'd into each scalar field. For instance, if one

[2]The current bag structure uses a sparse array with an index range from $0 \cdots 2^{32} - 1$. This restricts us to 32 bits of connection information. With the advent of IPv6, a much larger index range must be accommodated and work is under way to develop hash-based sets and bags. As a side effect, we will be able to develop connection structures with the full address information, thus removing the 32 bit limitation.

of the fields is the lower 16 bits of the source addresses in a subnet, the mask could be the higher 16 bit subnet mask of the address.

- The start and end times of the first and last record in the bag file.

The data table portion of the file consists of two columns. The first column is the 32-bit value that contains parts of the two scalar fields as specified by the user. The second column is a count of the number of flows, bytes, or packets for the flow records selected for the bag file.

The file is read and parsed by the application. Each 32-bit value is separated into its two components, OR'd with its corresponding mask value, and stored in separate integer variables in a data record object. Also stored in the object is the corresponding volume count. Each record is stored as an element in a time-ordered vector for rendering by the application.

4.3.2 Visual Structure

Figure 4.3 shows the FlowBundle visualization: a circle divided in half with 256 points around the top half and 256 points around the lower half. Each data point corresponds to an 8-bit portion of one of the scalar variables described above. FlowBundle is limited to 512 points to reduce clutter at smaller resolutions.

One of the advantages of using the 512 point layout on the circle is that a compact 256×256 matrix structure, M, can hold the connection data. The rows of the matrix represent indices to the points on the lower portion of the circle, while the columns represent indices to the points on the upper portion of the circle. Any element $M_{i,j}$ holds the traffic volume count for traffic between point i on the lower portion of the circle and point j on the upper portion. Cells with a zero value indicate no communication between the two points. This structure has several advantages. To render the visualization, the application only has to iterate through the matrix to determine which connections to represent. To determine to which connections a particular point i, j is related, only row i (if on the lower portion of the circle) or column j (if on the upper portion of the circle) must be iterated.

More information must be stored than just the volume information between connection endpoints. Each connection follows a (visual) path influenced by a set of

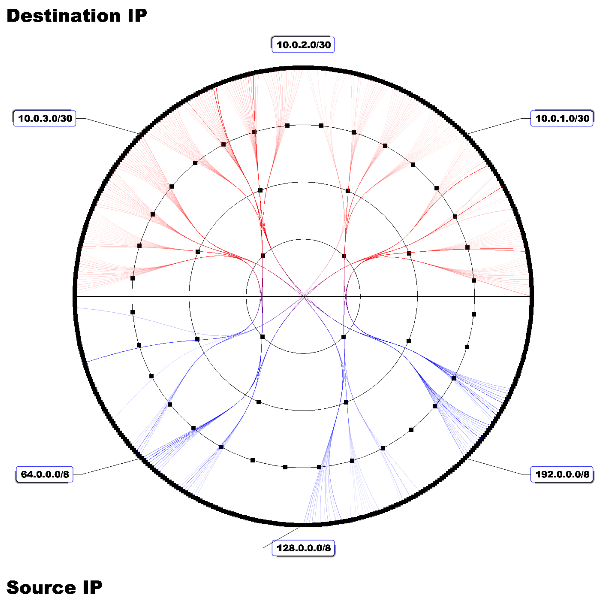


Figure 4.3: FlowBundle screen shot.

internal control points, and this (visual, non-network) route must be stored for rendering. Notice the three internal rings in Figure 4.3. The innermost ring has four equally spaced points around it; the next ring has eight points, while the final ring has 32 points. A combination of such points comprises the B-spline [4] control points for the connection. A B-spline is essentially a curve whose direction is influenced by the placement of a set of control points along the curve. These control points act like magnets which attract the curve. B-splines provides control over the direction of the curve to create the connection bundles. All the control point coordinates are organized in a radial tree structure. The four points in the innermost ring correspond to the four roots of the tree. Each root point on the inner circle is parent to two points on the second ring, which are in turn parents to eight nodes on the next ring, which are in turn parents to 16 nodes of the outer ring. This totals 512 nodes around the outer ring. This structure provides a convenient method for generating a route between the two endpoints of a connection. With two points on the outer circle, the tree is traversed from the outer points to the inner circle by going from parent to parent. Once the traversal reaches the innermost ring, a full connection path of points is created. To store all of the connections, a 256×256 matrix, M is used. In this case, any entry $M_{i,j}$ contains a connection path if points i and j are connected. This configuration ensures that path connections are calculated only once.

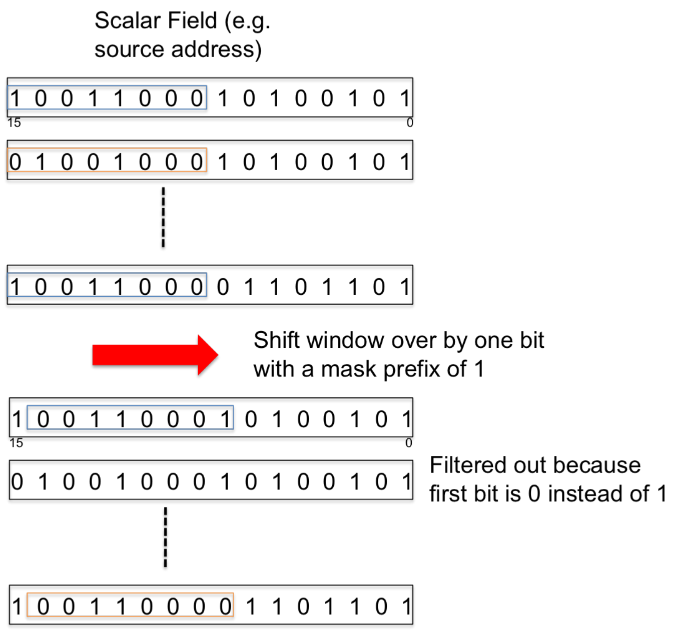


Figure 4.4: Example of a sliding bit window.

This begs the question: How does one process the list of data records discussed in the previous section into the 256×256 matrix described above? As mentioned above, each data record is composed of two scalar fields (the total length of which adds to 32 bits). In order to reduce each field to the eight bits required for the 256 points on the circle, the eight most significant bits for each scalar are used as the indices to the matrix. Figure 4.4 shows a 16 bit scalar field with the eight most significant bits (MSBs) highlighted in a smaller rectangle. Records with the same eight MSBs have their volume counts added together in the matrix structure. This aggregates the values by their eight MSBs. The drill down feature of the application slides the 8-bit window down the scalar field thus generating a new matrix. As the 8-bit window is slid down the scalar field, a prefix is created with the upper bits and is stored along with the matrix in order to maintain location information within the drill down. In the example shown in Figure 4.4, the 8-bit window is slid down by one bit and the prefix mask is set to one. As a result, any records that have a prefix of zero are ignored, and the remaining records are aggregated based on their next eight bits.

As an abstraction, the 8-bit sliding window might not be appropriate for novice analysts. Therefore, the internal radial tree layout is used as the drill down mechanism to provide the mask and perform the bit sliding. The tree structure allows for drill down on a certain section of the circle simply by clicking on any of the internal parent nodes as shown in Figure 4.5. Each node has an associated mask and sliding bit length that is applied to the data when it is selected in drill down mode. To drill down on a smaller section of the circle, the user clicks on a parent node closer to the leaf. Clicking on a leaf node will slide the bit window by eight bits and provide a mask equal to the leaf number. To drill down on a larger section, the user clicks on nodes closer to the root. In Figure 4.5, the node highlighted is associated with the mask 000 and a sliding bit length of three. Note that there are 8 nodes at this level, as can be represented by 3 bits.

As mentioned in Section 3.2, FlowBundle uses a tabbing mechanism when the user executes the drill down feature. Each drill down visualization is either launched to replace an existing visualization or into its own tab (as chosen by the user). As a result, each FlowBundle tab has its own matrix structure. When drilling down, a new tab is created along with a new Matrix M.

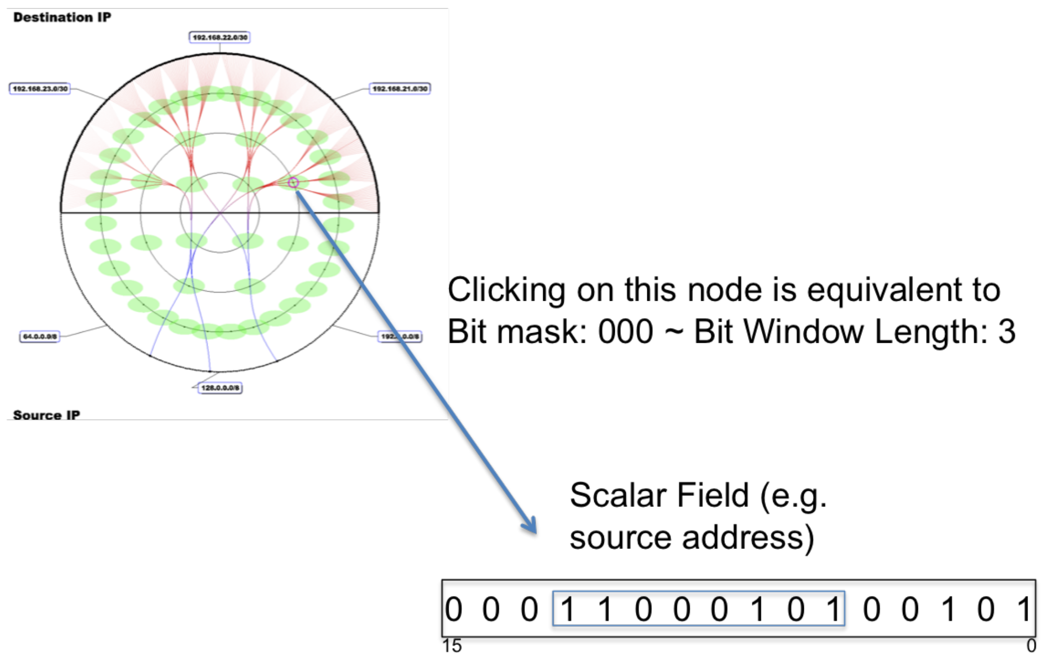


Figure 4.5: Mapping of the radial tree structure to a bit window.

As mentioned in Section 3.2.1, the visualization contains a β-slider (beta slider) that controls the tightness of the bundles. The formula for loosening the bundles is outlined in the Hierarchical Bundling paper [17]. The algorithm straightens the B-spline curves by bringing the control points closer to the first point in the connection; this occurs as β approaches zero. The formula for calculating the location of the control points is as follows:

$$P'_i = \beta * P_i + (1-\beta)(P_0 + \frac{i}{(N-1)}(P_{N-1} - P_0)) \tag{4.1}$$

where N is the number of control points, i is the control point index, β is the bundle strength in the interval $[0,1]$, and $P_i \in \mathbf{P}$, where $\mathbf{P}$ is the set of control points. The control points are recalculated on each refresh of the screen to support a seamless transition as the β-slider is moved.

4.3.3 View

The rendering mechanism for the FlowBundle diagram is organized as shown in Figure 4.6. The three main classes responsible for rendering are Canvas, Display, and Layout. The main panel handles most of the interaction with the user, including retrieval of information regarding input data. The panel then interacts with the DataController to load the DataTable, which is subsequently passed to a Layout object for rendering.

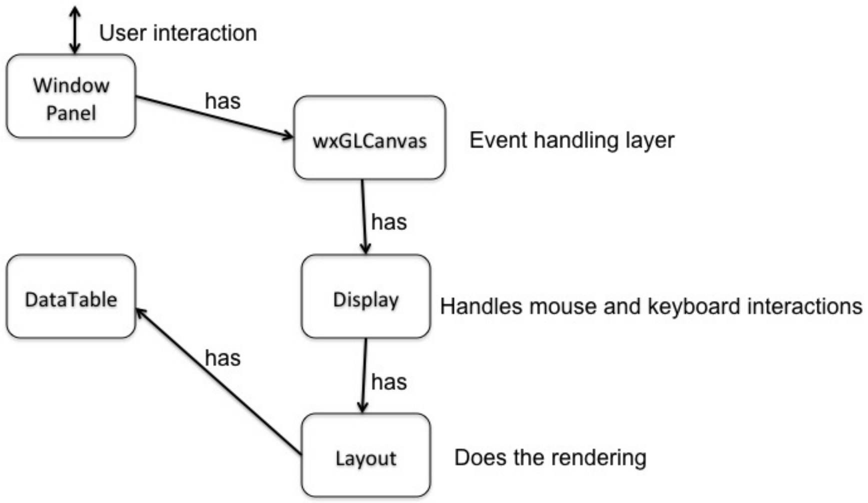


Figure 4.6: FlowBundle rendering structure.

The Canvas class is a subclass of wxGLCanvas (wxWidget's OpenGL interface).

It handles all WX events (*e.g.*, mouse and keyboard) that are directed towards the interface. These events are then passed to a Display object, which has functions for all the mouse and keyboard events. This was designed so as to separate the main OpenGL code from the windowing system code, in an effort to support portability. Since the code is separate, other windowing systems can be used instead if desired. Furthermore, the Display object also allows for swapping different visualizations between Canvases. The Layout class contains all of the code that renders directly to the screen using the OpenGL interface.

The current Layout object also contains the radial tree structure (with point coordinates), in addition to the current DataTable and connection matrix. As a result, the Layout object iterates through the radial structure in order to setup the visualization topology. It then iterates through the connection matrix to visualize the connection data. Since all the point coordinate data is stored in the radial tree structure, modifications to the coordinates on the structure affect change on the connections. This is done, for instance, when applying the linear distortion on the nodes along the outer circle.

4.4 NetBytes Viewer

NetBytes Viewer is a visualization tool that presents the traffic volumes to or from an individual entity (host or subnet) on a network. It utilizes a 3D impulse graph having the port/protocol number along the x-axis (horizontal), time along the z-axis (horizontal), and volume (flows, bytes, packets) along the y-axis (vertical). This allows the user to view the hourly port/protocol activity of a host or network over time. With the goal of addressing some of the common issues associated with 3D displays, two smaller 2D displays are available that render the time versus volume and the port/protocol versus volume axes individually for selected slices of the 3D graph. Interactive features allow the user to select specific ports or times to visualize with the the 2D plots. Other features include adjusting the ranges for each axis, and rotating the plot.

This section outlines some of the key implementation decisions for NetBytes Viewer. These include transforming the SiLK data and choosing a suitable database for storage.

4.4.1 Data Table

The data input files for the application are port or protocol bags. In this case, the NetFlow data is filtered by the IP address of a single host, or by a subnet. The number of flows, bytes, or packets are counted on an hourly basis over a period of time on a per port/protocol basis. Since three dimensions (time, ports/protocols, and volumes) are displayed, a data store that provides more powerful queries than available with a flat file is required. As a result, various database systems were evaluated to determine their suitability for this visualization application. The key issue is that of scalability, since gigabytes of NetFlow data can be collected per day for a large network. This section will outline the different databases that were considered. The goal was to find a database that had the following characteristics:

1. Allows for powerful queries and sorting on different columns.
2. Retrieval times on two months of our test data on the order of seconds. Two months was chosen because longer time periods may not render as well under an hour granularity.
3. Quick loading times (millions of records on the order of seconds).
4. High scalability and availability.

4.4.1.1 OLAP Cube

An Online Analytical Processing (OLAP) cube can be thought of as a generalization of a spreadsheet (or 2 dimensional array) in any number of dimensions. It also has the ability to create hierarchies on the dimensions to enable aggregations or rollups of the data [32]. For example, if there is a time dimension that is at an hour granularity, that dimension can be rolled up into days, months, quarters, or years. The multiple dimensions allow data to be ordered by any number of dimensions, *e.g.*, by time, by port, by IP address. This allows for highly flexible queries on the data. The OLAP cube's main advantage over a relational database is its ability to perform instantaneous analysis of the data, since the data is formatted in sorted arrays that facilitate quick aggregations or drill downs [32]. However, in order to gain flexibility, the OLAP cube performs numerous pre-aggregation calculations, which results in greater disk storage requirements (due to redundant data) than a typical relational database. Further, building and updating the cube requires additional computation time. As a result, writes to the cube are not updated in real time, but rather the cube is optimized for reads. Thus, the system is not useful for constant updates, but rather batch updates at low usage times. This set of constraints are limiting for forensic network analysis as new data for an interesting host cannot be fetched without updating the cube. In such a case, we might need to update the cube hourly, or have a staging database that contains the most current data for the day and that gets processed into the cube on a nightly basis. Thus, there is a tradeoff between these issues and the flexibility of the OLAP cube.

Microsoft SQL Server 2005 Analysis Services[3] was used for the OLAP cube and a star schema for the underlying database structure was designed, as shown in Figure 4.7. This figure demonstrates a set of different dimensions that are used to select, group and aggregate data at the desired level of detail [32]. For NetBytes Viewer, the capability to group data by time, protocol, flow volume type (*i.e.*, flows, bytes, packets), IP-port combinations, and data flow direction was deemed necessary. Notice the star-like topology of the tables in the figure — hence the term star schema. This configuration allows for queries on any of these groups as well as the creation of useful aggregations.

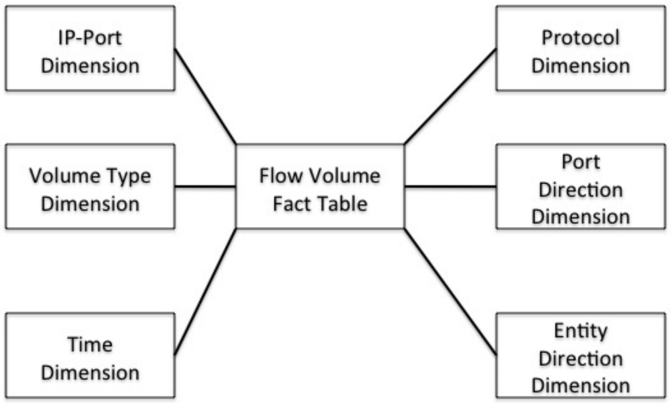


Figure 4.7: OLAP star schema.

The fact table represents the subject of the analysis [32]. NetBytes Viewer is concerned with traffic volumes. As a result, the fact table contains volumes and links

[3]`http://www.microsoft.com/Sqlserver/2005/en/us/default.aspx`

to the other dimensions. Combinations of these dimensions are used to make queries on the data in the fact table. The query language used by SQL Server for its Analysis Services is called MDX (multidimensional expressions). MDX is query language that is well suited for multidimensional queries and meets our stated requirements.

The Microsoft SQL Server version of OLAP has limitations that result in scalability issues (both in terms of storage and computation time). The main issue with Analysis Services is that it uses an open standard called XMLA[4] as its client query protocol. XMLA is an encoding format that stores meta information about the data, which works well for small datasets. When transferring large amounts of data, XMLA significantly inflates the data size (in some cases, it can more than double the size of the data). Furthermore, all the data must be converted from float and integer data to ASCII to be used with XML, which means that much of the data processing time is tied up in the serialization of the data.

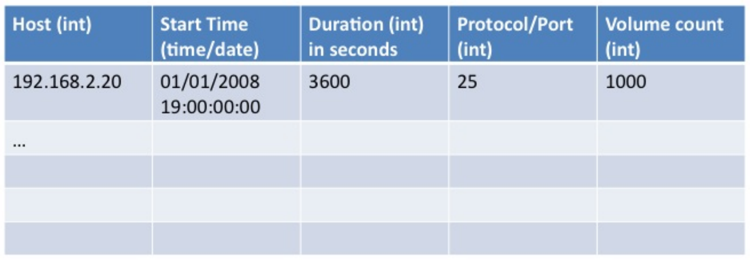
To test the performance of the Microsoft SQL Server, data was loaded from a machine that was infected by a game server. The game server generated over three million data points within a single month and provided a benchmark for performance testing. A single query on the OLAP cube required 15 minutes to return the data to the client visualization. The SQL Server query tracer showed that the time required to serialize the data into XMLA was the bottleneck. Microsoft designed its platform for the business sector, who often need to be able to quickly analyze and aggregate their data, but do not necessarily need to return large amounts back to the user. With XMLA, they can capitalize on XML's ability to easily integrate data with view while sacrificing the ability to move large amounts of data around quickly.

As other OLAP cube platforms may not have the same limitations as Microsoft's SQL Server, the OLAP cube remains an interesting and viable option for the data problem facing the network security community.

[4]`http://www.xmlforanalysis.com/`

4.4.1.2 Relational Database

A relational database is an all purpose (online transaction processing) database. It is known for its simplicity and efficiency. It is not optimized for either reading or writing; but rather, to do both equally well. The application was tested using MySQL[5] 5.1 because it is a full featured, open source database. This version contains new features designed to support large datasets, such as table partitioning and sub-partitioning. Table partitioning is a table organization scheme whereby data is stored into different table files based on the values of one or more of the table's columns. At a virtual level, the table is seen as a single entity, but underneath it can be spread across several storage devices. This allows queries to run faster because the database does not have to search partitions that are not within the column value ranges of certain queries. Furthermore, with the table being stored across multiple devices, the database can take advantage of parallelism to decrease retrieval times. Lastly, partitions keep files smaller, thereby reducing seek time and the potential for file corruption. Partitions can be determined by column value ranges, enumerations or hash functions. MySQL builds on simple partitioning by further allowing sub-partitioning on individual partitions.



Host (int)	Start Time (time/date)	Duration (int) in seconds	Protocol/Port (int)	Volume count (int)
192.168.2.20	01/01/2008 19:00:00:00	3600	25	1000
...				

Figure 4.8: Relational database table schema.

A database using the table schema shown in Figure 4.8 was designed and implemented. This table contains a host IP (or subnet IP), a start time and duration for the record, the port or protocol number for which the traffic occurred and the volume count. Other information was encoded in the table name rather than in the data itself to limit the amount of data stored in the table, providing another layer of data partitioning. Information included in the table name is as follows:

[5]http://www.mysql.com/

1. Traffic flow direction (data going *to* or coming *from* the host),
2. Port direction (source port or destination port),
3. Protocol type (ICMP, TCP, UDP, or other[6]), and
4. Volume type (flows, bytes, or packets).

The table schema is kept simple to eliminate complex joins that could slow down retrieval times. Although the batch data loading tool for MySQL is quick, it is limited in that it can only load batch data into a single table at a time. Therefore, it is easier to use simple tables to strike a balance between efficient data loading times and quick reading times. Each table is partitioned into 12 monthly partitions based on the start time, so that older data can be dropped and/or archived more easily. Also, each monthly partition is broken into 64 sub-partitions using a basic modulo 64 hash function on the host column in order to ensure that host data for IPs in a subnet are dispersed evenly to keep table files smaller. Indexes were created on the host, start time, and port/protocol fields to help limit the need for table scanning.

Using the same test data as for the OLAP cube, the query performance time was approximately 30 seconds — a significant improvement over Microsoft SQL Server Analysis services. Such performance was deemed acceptable as the amount of data retrieved is high and the database was still able to return the data in a reasonable time frame. In the future, continued testing of the database's performance with high data volumes is required. Furthermore, MySQL server clustering will be investigated. Clustering allows data to be transparently distributed across several nodes, so that the data is partitioned across machines rather than storage devices. As the amount of data increases, such a configuration could prove necessary.

4.4.2 Visual Structure

An interesting aspect of OLAP cube is how it returns data. Data is returned sorted in a multidimensional array structure with time along one axis and ports along the other. This type of structure makes it easier to slice and dice the data from the 3D impulse graph to the 2D visualizations. As a result, a multidimensional linked

[6]This is only relevant for port data.

list data structure (shown in Figure 4.9) was developed as the underlying storage object for NetBytes Viewer. When a query is made on the database, it is given a lower and upper bound on the time stamp as well as a lower and upper bound on the port/protocol number. The data is loaded into the linked list structure, with the time stamps placed in order along the top of the structure and the port/protocol numbers in order along the side of the structure. Only time stamps and port numbers that have data points are present in the structure. Iterating through a time stamp gives the volume amount for that time period. Each internal cell with data has a pointer to its port/protocol number as well as its time stamp.

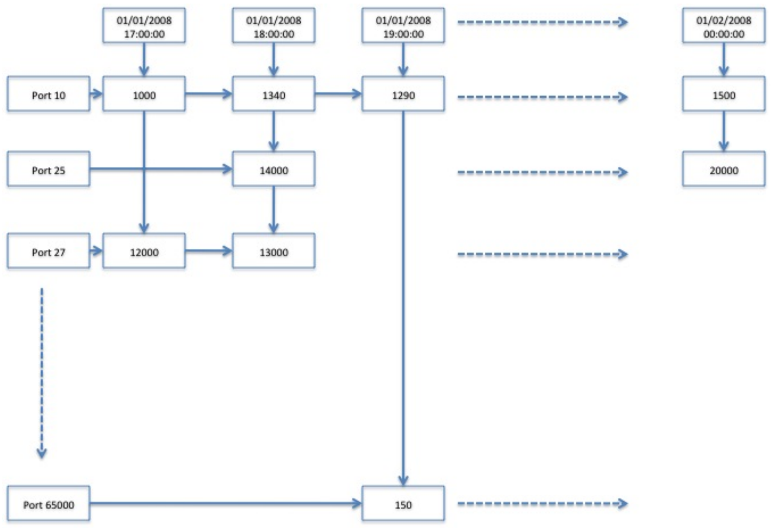


Figure 4.9: Multidimensional linked list. Port information on the vertical axis. Time information on the horizontal axis.

To display the data in the impulse graph, the entire structure is iterated through by port/protocol number, changing the color with each change in port number. The time and port number are mapped to the 3D impulse graph by subtracting the starting time (in epoch time) and the starting port number from each entry in the structure. Such a structure can be used to search for the next port or time stamp that contains data, which is required for the 2D renderings (time versus volume and port versus volume) of the 3D data. For these, the port index and the time index of the data are passed to the 2D display, which iterates through the linked list of data points.

4.4.3 View

The rendering mechanism for NetBytes Viewer is organized similarly to FlowBundle, but with a few different features (see Figure 4.10). The main panel has three rendering windows (one main window for the 3D view and two smaller 2D screens). The displays may be swapped so that any of the windows can be switched into the main view for more detailed analysis. To handle this feature, a class called SwitchCanvas was created. A SwitchCanvas may contain a generic Display object. When swapping the views, the Display objects are swapped amongst the SwitchCanvases and the views re-rendered.

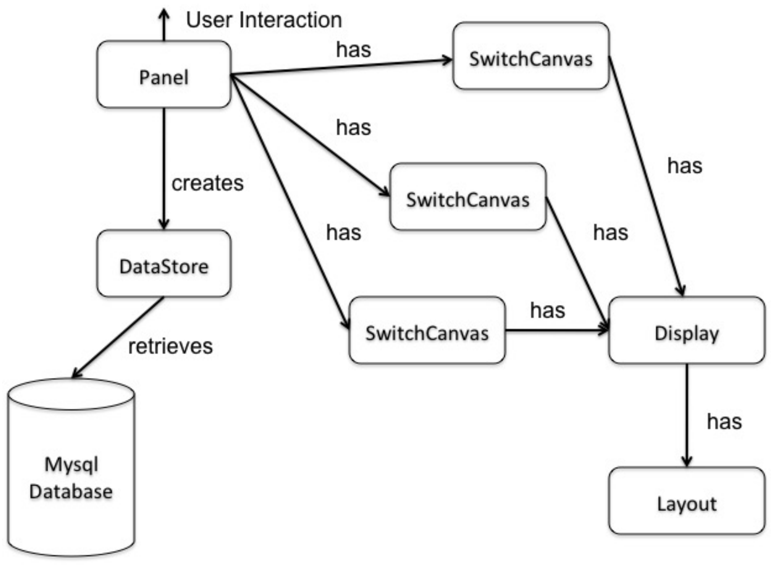


Figure 4.10: NetBytes Viewer rendering structure.

As with the FlowBundle rendering organization, the Display object handles all of the mouse and keyboard events from the user, passing control to the Layout object to perform the rendering (using OpenGL).

Chapter 5

Results and Evaluation

In this chapter, an evaluation of the FloVis framework is discussed. The evaluation process took two forms. First, FloVis was used in several case studies to find three separate suspected intrusions to establish that the visualizations are useful for providing insight into the happenings on a network. Second, a FloVis demo application was created and demonstrated to 12 trained security analysts and researchers in the network security industry in order to solicit feedback and ideas for improvement. The following sections describes these endeavours in more detail.

5.1 Using FloVis to Monitor the Network

In this section, FloVis is evaluated using data from a small private network, consisting of four /24 subnets. It was collected with the SiLK collection tools from February 2006 to March 2007. The network has 1024 addresses and approximately 100 active hosts. This network had not been analyzed with the SiLK tools before this capture time; it contains some interesting patterns that signify game server and scanning worm activities. At the time of collection, the collector supported 1 second time resolution and the data was collected from a Cisco router that did not support enhancements (such as first packet TCP flags). This limits some of our analysis capabilities for this dataset.

The SiLK collection tools were installed on the network's border with the Internet so that all traffic between internal and external hosts can be seen. A unique attribute of the network in question is that there are machines from several different companies within it and there is no single administration that oversees all the machines. The result is that some machines may become infected on the network, but not be examined and quarantined. In this section, the following case studies are investigated: a host acting as a game server, an intrusion from a scanner, and a machine that becomes a scanner.

5.1.1 Host Infected by a Game Server

NetFlow collection started on the network in February 2006 and a forensic network analysis began shortly thereafter. One host (shown in Figure 5.1(a)) in particular exhibited interesting behaviour and was loaded into the NetBytes Viewer. Analysis suggests the machine had been hijacked. As seen in the Figure, there is a significant amount of UDP traffic on ports 27015 (brown band in Figure 5.1(a) and the subject of Figure 5.1(b)) and 26900 (yellow band in Figure 5.1(a) and the subject of Figure 5.1(c)) after using the NetBytes Viewer's port selector mode (green highlight in Figure 5.1(a)). These ports are known service ports used by a Half-Life game server[1]. This machine had been compromised and was being used as a game server.

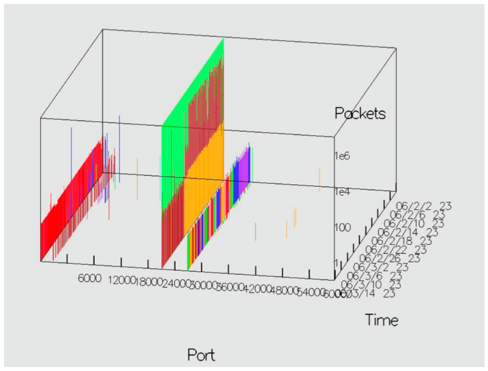
Also of interest in Figure 5.1(a) is the consistent traffic on port 123 (the red strip in the diagram). This port is typically used for NTP (Network Time Protocol) which suggests that the host machine was likely intended to be a time server for the network. NetBytes Viewer makes these patterns easy to see without scanning through text-based SiLK records.

UDP traffic coming from the host is shown in Figure 5.2. There is ephemeral port activity as well as two prevalent bands of consistent data flow. One data flow band is on port 27,014 (pink band that is highlighted by the green port selector) and is likely the outgoing traffic for the game server running on the machine. The brown band (long band on the right hand side of the diagram) maps to port 54,020 which seems to be typically used for client applications, such as FTP and messaging, on Linux servers.

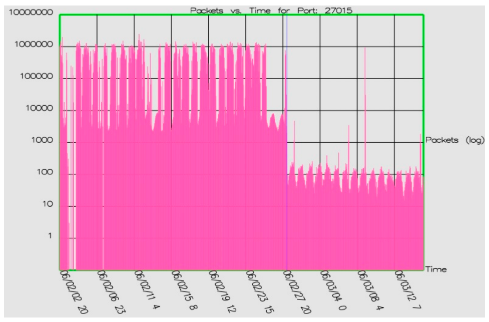
5.1.2 Infected by a Scanner

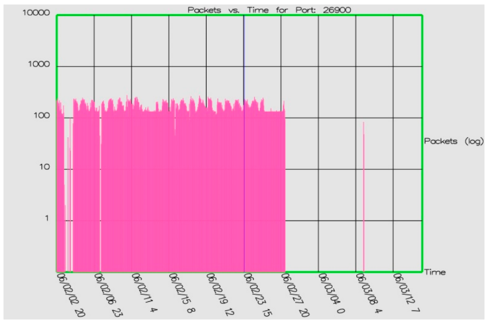
In this section, the combination of FlowBundle and NetBytes Viewer are used to find a host infected by a scanner. Scanners are hosts designed to probe other machines on the network in order to learn which services the machines are running. Scanners do this by sending packets on specified ports and waiting for responses. From a response, the scanner can determine which hosts are active and which services they offer. Once particular services are identified, the scanner may try to compromise vulnerabilities in those services to gain control over the machine [7, p. 44].

[1]`http://www.iss.net/security_center/advice/Exploits/Ports/27015/default.htm`



(a) NetBytes Viewer showing some unusual behaviour on ports 26900, and 27015.





(b) A closer look at the UDP traffic on port 27015.

(c) A closer look at the UDP traffic on port 26900.

Figure 5.1: NetBytes Viewer shows unusual UDP behaviour going towards the host 5.1(a), with 2D views of port 27015 5.1(b) and port 26900 5.1(c).

The key for finding intrusions on a network is to have a starting point for analysis. In other words, with this huge amount of data, how does one begin to find intrusive behaviour? This is an important aspect of network security analysis, and there is a constant search for new ways to analyze data to find such starting points. In this scenario, the paradigm is to search for external hosts that are scanning an internal network to find intrusions. Detecting scans is not particularly interesting because most scans yield little in terms of responses from internal hosts. However, those hosts that respond to the scanner are of interest because they are good targets for intrusion. Using the SiLK tools, along with a hash table plugin, all the unique internal hosts that were contacted by external hosts were counted on a daily basis for the period

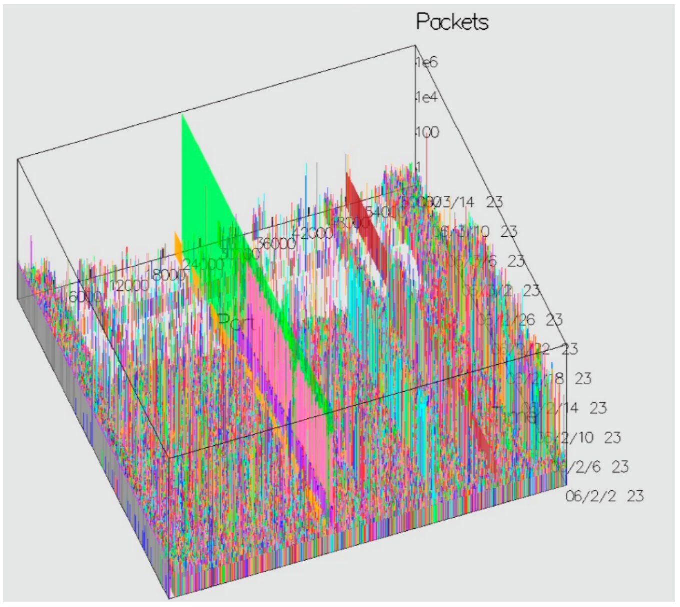


Figure 5.2: UDP traffic coming from the game server.

of June and July 2006. Since no more than 100 hosts existed in the internal address space, any host that tried to contact more than a few hundred hosts was labeled as a scanner. This analysis yielded 14 distinct scanning sources. Daily connection bags were created for each of the scanners to see who they were communicating with in the internal network and then visualized using FlowBundle. One such visualization is shown in Figure 5.3. This diagram shows all the scanning activity for June 2nd, 2006. The scan is characterized by the large degree of fan out from the single scanning source. Most of the connections are faint, indicating a single connection attempt; however, a single group, in the top right corner is more opaque, indicating multiple connections or higher data transfer between hosts. In this scenario, it might indicate that the scanner is uploading an intrusive payload to the host or it could indicate multiple connection attempts. In this case, it was the latter.

After investigating other days in June, a suspicious interaction was found on June 29th, 2006 which is shown in Figure 5.4(a). On this day, there were three scanners but only one connection (on the upper right corner of the diagram) saw a significant amount of data transfer. Using the drill down mode, described in Section 3.2.1 and shown in Figure 5.4(b), it was discovered that the connection goes to host 192.168.20.163 as shown in Figure 5.4(c). Loosening the bundle, as in Figure 5.4(d), shows where the connection originated (in this case, 123.99.63.81).

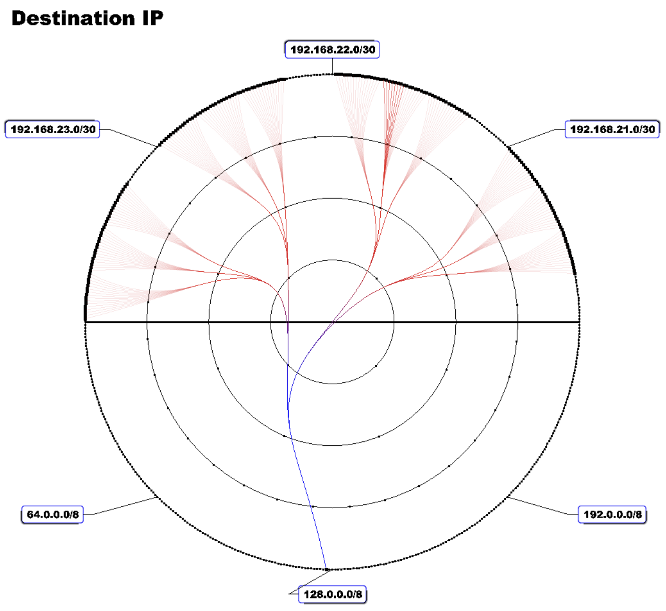
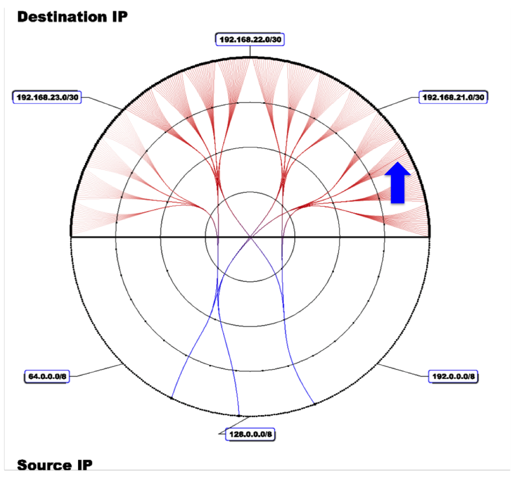


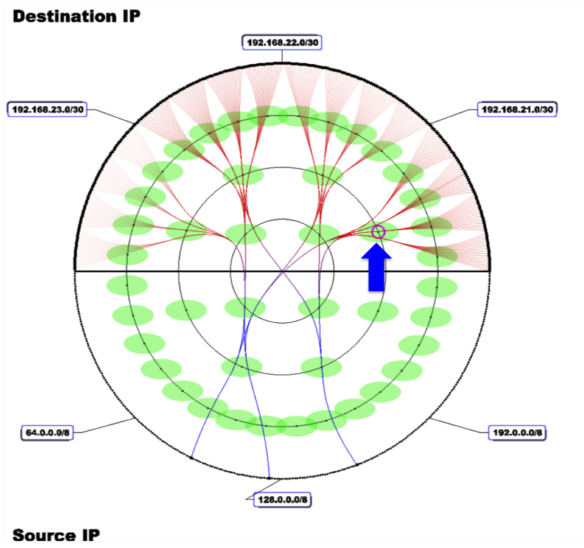
Figure 5.3: FlowBundle showing a scanner on June 2nd, 2006.

Figure 5.5 shows the bidirectional FlowBundle view of the three scanners. The bottom portion of the circle shows the external-to-internal connections from the scanners, while the top portion shows the internal-to-external host responses. Notice that while the scanners scanned the entire subnet, only about a dozen hosts responded. Furthermore, the single dark connection appears on both the top and bottom portion of the circle as it did in the unidirectional FlowBundle.

Next, the NetBytes Viewer can be used to visualize the port interactions between the two hosts to see how they have been communicating. NetBytes Viewer is loaded with the port activity data for host 192.168.20.163. Selecting TCP traffic to destination ports on this host for June and July 2006 triggers the display shown in Figure 5.6(a). Using the selection and adjust axis features, a spike in data on port 3306 is apparent. This activity started on June 29th (the same day of the data transfer from the scanner) and ended on July 10th. Looking at UDP data transfer to the host destination ports, as seen in Figure 5.6(b), another interesting spike is observed. This spike is on port 137 and occurs between June 29th and July 10th (the same length of time as the TCP spike). Some investigation on the Internet for ports 3306 and 137 indicates the likely scenario. Port 3306 is a popular port for communication with a MySQL database and port 137 is used for NetBios communications on Microsoft

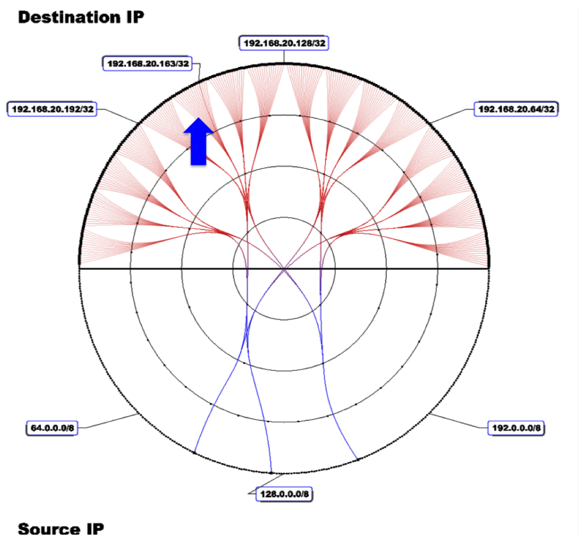


(a) Bundle diagram of 3 scanners with one intrusion. June 29th, 2006.

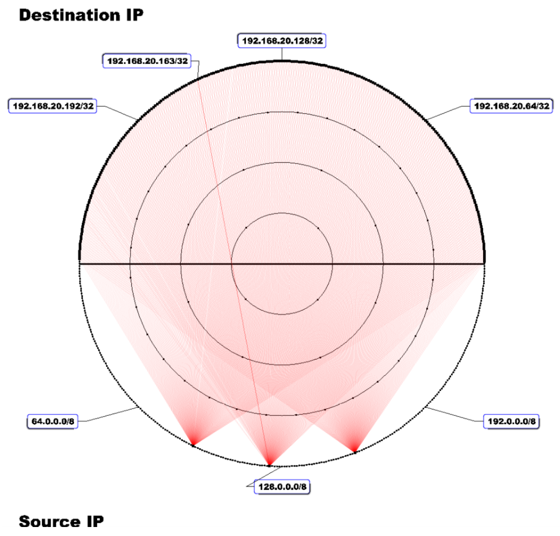


(b) Drilling down to investigate the opaque connection on the right.

Figure 5.4: Investigating scanning activity on June 29th, 2006.



(c) A label shows that the offending connection goes to host 192.168.20.163.



(d) Loosening the bundle shows where the connection originates (123.99.63.81).

Figure 5.4: Investigating scanning activity on June 29th, 2006 (cont.).

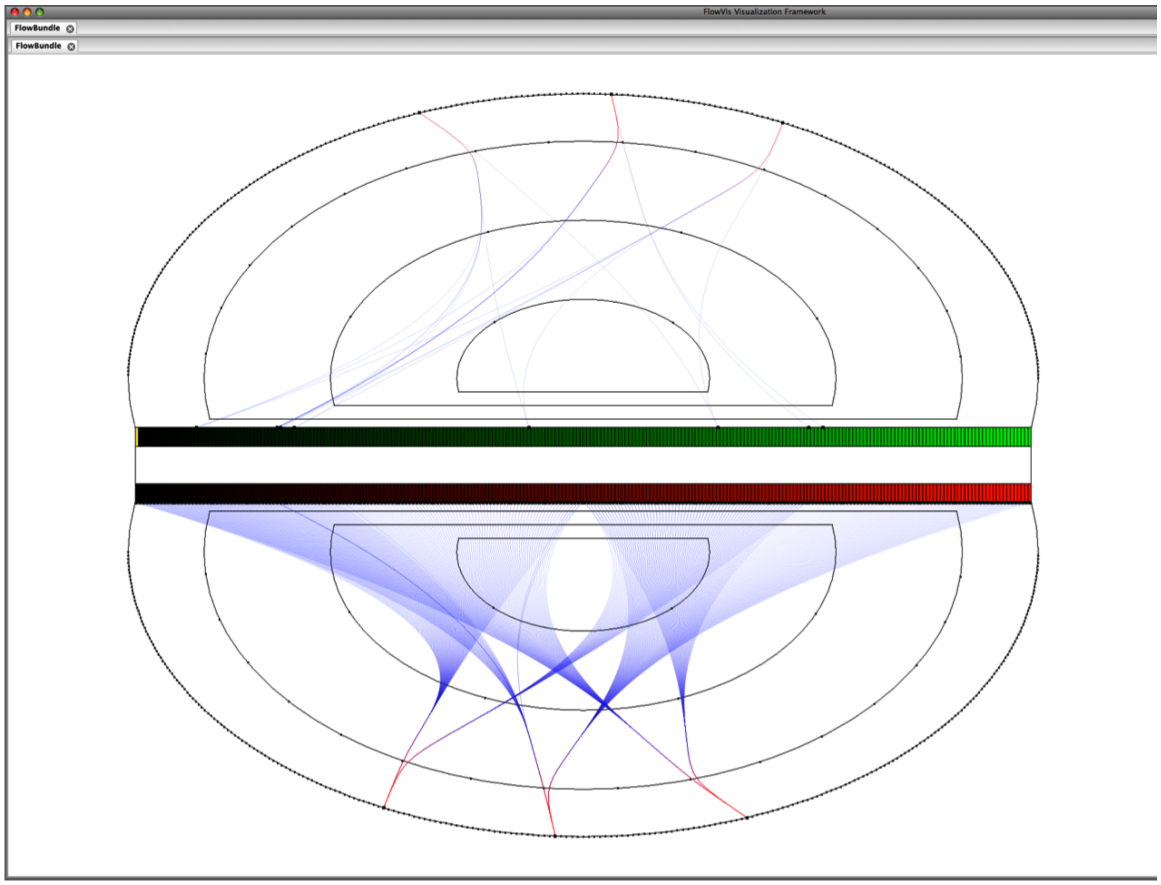
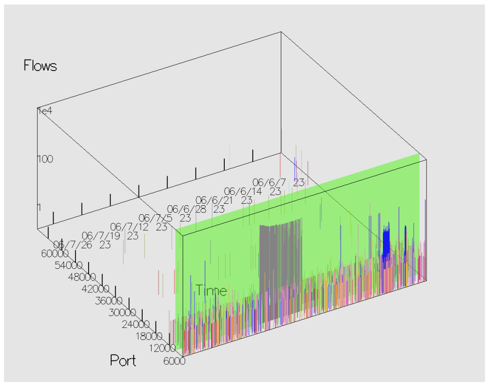
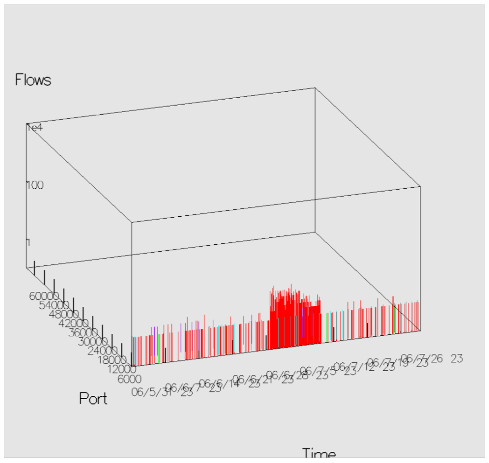


Figure 5.5: Bidirectional view of the three scanners on June 29th, 2006. The top portion of the circle is internal-to-external host connections while the bottom portion is external-to-internal connections. Notice that the scanners scan the entire internal network but only a few hosts respond.

hosts. MySQL systems with weak passwords can be susceptible to a MySQL botnet attack where the bot tries to login to obtain root access to the server. If it succeeds, it installs malicious code on the server. There is also some indication that port 3306 (TCP) may be used by the eMule or eDonkey peer-to-peer applications as well. Port 137 may be a cover for transferring some sort of data between the scanner and the local host. Without the payload packet information, it is difficult to determine exactly what was transferred; however, this machine has exhibited behaviour that may warrant the investigation of a security administrator. Further investigation of the port activity on the host shows significant traffic on ports 80 and 443, consistent with the activities of a web server supported by a MySQL server to facilitate transactions for the site.



(a) Activity on port 3306 between June 29th, and July 10th.

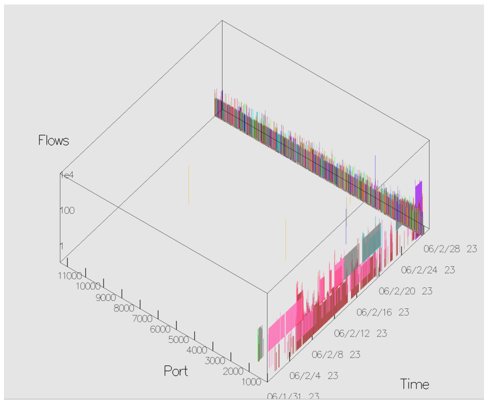


(b) NetBytes Viewer for host 192.168.20.163 filtered by UDP (data to host with destination ports).

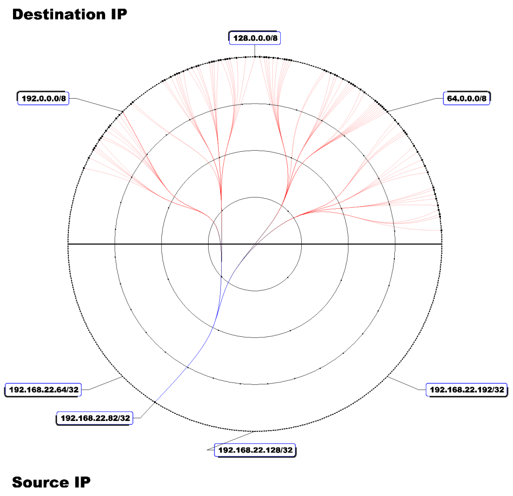
Figure 5.6: Showing the traffic volumes in the NetBytes Viewer for June and July 2006.

5.1.3 Infected Host Becomes a Scanner

Another incident of infection occurred on February 28th, 2006 at 17:00 to host 192.168.22.82. The intrusion was spotted originally using another visualization which is part of the FloVis Framework called the Activity Viewer [40] (which is outside the scope of this thesis). In the case of host 192.168.22.82, on the day in question, the machine started exhibiting behaviour whereby it was using the same ports to handle both client and server traffic, which was different behaviour from what had been observed previously. To analyze further, the host was loaded in the NetBytes Viewer for the time period from February 1st to March 1st. Figure 5.7(a) shows the TCP destination port traffic coming from the host. The colourful band along the back portion of the 3D graph indicates the host started to port scan on February 28th at 17:00. Figure 5.7(b) shows the host connection patterns for the infected host for the entire day of February 28th, 2006. As illustrated in the graph, the host is scanning almost the entire network space. The dark red line on the top left hand portion of the circle indicates the host conducted internal scanning as well. Figure 5.7(c) shows the ports host 192.168.22.82 communicated over on February 28th. This figure shows the flexibility of FlowBundle in matching different NetFlow attributes in the visualization. The host scanned every port.

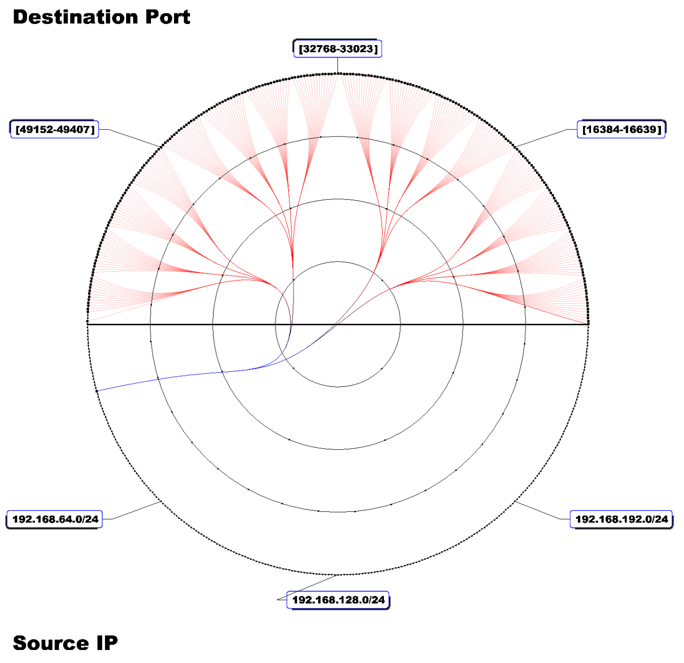


(a) Infected host shown in NetBytes Viewer. The diagram shows destination port traffic coming from the infected host.



(b) Network scanning from infected host for day February 28th, 2006.

Figure 5.7: Infected host conducts horizontal and vertical scans on February 28th, 2006.



(c) FlowBundle shows the relationship between the infected host and all destination ports communicated with on February 28th, 2006.

Figure 5.7: Infected host conducts horizontal and vertical scans on February 28th, 2006 (cont.).

5.2 User Evaluation and Feedback

User input is an important part of evaluating the usefulness of FloVis. However, feedback from qualified individuals (the target audience for FloVis) is the most valuable. As a result, a series of demonstrations were conducted and comments were solicited at Flocon 2009[2] in Scottsdale Arizona. Flocon is a conference directed towards network analysts and researchers (both academia and industry) interested in security analysis of network traffic using NetFlow. Attendees included security analysts from CERT, US CERT, the US Department of Homeland Security and the US Department of Energy; visualization researchers were also in attendance from Stanford University, Carnegie Mellon University and Secure Decisions, as well as researchers and technical resources from several other organizations including TARA and Norwegian CERT.

As part of the demonstration, FloVis researchers accompanied conference participants for a hands on demo of FloVis on a laptop using a semi-guided tutorial. The tutorial led the users through the Activity Viewer, FlowBundle, NetBytes Viewer and the SiLK query tool using data files from the weakly anonymized data set described in Section 2.1. During the hands on demo, participant feedback and new feature suggestions were recorded.

5.2.1 Overall FloVis Framework

The feedback for the FloVis framework was mainly positive. Analysts supported the idea of having one application that is extensible in which additional visualizations could be added on an as needed basis instead of requiring several applications to evaluate data. Analysts also expressed a desire for a historical trail of how they proceeded through their analysis. They saw the tabbing drill down option in FloVis as helping in that regard as it made it easier for them to backtrack in their analysis.

Analysts also appreciated that the framework was written using a platform independent windowing system. A request for application ports to the Microsoft Windows environment was made as this seems to be a popular operating system for client machines.

Evaluators were also pleased that FloVis did not use SiLK data files directly but

[2]`http://www.cert.org/flocon`

rather an intermediate data source. This allows FloVis to be used with analysis tools other than SiLK.

Another idea of key importance to the analysts was that the backend data used to generate the visualization was not lost but rather travelled along with each visualization presented to the user so that he could access this at any time. This is addressed in FloVis by passing a set of parameters between visualizations which can be used to make underlying SiLK queries. In this way, the SiLK query is always preserved. However, portions of the query may be lost as some visualizations do not use some parameters. As such, FloVis needs a generic data structure that can house all SiLK query parameters so no data is lost between visualizations. Such a structure would also allow any plugin to launch any other plugin and would be a valuable addition to the framework.

The analysts also talked in great detail about the need for data fusion — importing data from different sources into the framework. They stated that using NetFlow data alone is not sufficient to find problems. They want to be able to cross reference flow data with DNS, AS, and geographical data to provide more context. DNS provides significant information about an IP address as it can provide the organization to which an IP address belongs, which can help in determining whether the host is intrusive. Geographical information can be important because, for instance, the US government might want to know if a host connecting to one of its networks is from a country looking to gain access to key intelligence. Geographical information is also important when conveying information up the chain of command. Management does not usually have the technical skills to understand networks and IP addresses but they do understand locations; therefore, maps are a powerful means to communicate. Contextual information can also be retrieved about hosts by doing a web search on an IP address, which may provide data about infections, domain registrations, providers and ownerships.

The ability to annotate visualizations was another feature of importance identified by the analysts. They want to be able to make notes about hosts and networks and then retrieve that information or send it along to other analysts. Also, the analyst wants to be able to save visualizations. Currently, there are no provisions in the framework for annotation, although it has been identified as a need. This might involve the creation of a user session or work project file to allow analysis to be tied to a user and saved in a globally accessible database. FloVis could be integrated with some sort of knowledge management website or wikipedia to make collaborative analysis easier.

Analysts also stated that they have different lists of IP addresses. For example, they have a list of bad IP addresses, a list of potentially bad IP addresses, and a list of safe IP addresses. They also have network topology information (in some cases incomplete); therefore, a visualization focusing on grouping hosts in various ways could be an important organizational tool for any analysis.

Lastly, some analysts showed concerns about the scalability of the application. US CERT, for instance, retrieves over one billion SiLK records per day. As a result, data reduction is important, but data must be reduced in such a way so that unusual behaviour is not "reduced away". They thought that using SiLK bag files was a good way to reduce the dataset but wondered if FloVis could handle the rigorous data volumes associated with such a high data load. Currently, the FloVis team does not have access to a significantly large dataset to do such testing, but have had a few offers from analysts who have access to such data to test the system.

5.2.2 FlowBundle

FlowBundle received some encouraging feedback from the analysts at Flocon. Many commented that they liked that FlowBundle was a *simple* graph with only 512 points along the outside of the circle. They did not want something that showed hundreds of thousands or millions of points (hosts) as it was too overwhelming and would be hard to analyze. Furthermore, analysts enjoyed the flexibility offered by the application in that it could handle any attribute pairing within a NetFlow record. This allows the user to visualize interactions not only between hosts but also host to port, host to protocol, port to port, protocol to port, etc.

During the demo, there were several inquiries about the use of colour on the bundles. Many analysts thought colour could be used for more than just direction. Some thoughts included using colour to indicate TCP flags. Some analysts were also concerned whether FlowBundle would be useful in analyzing hosts that are communicating with hundreds or thousands of hosts or under circumstances where there are heavy traffic loads. As a result, filtering and bundle selection were seen as key feature additions. Such features would allow the user to identify individual connections more easily by manipulating bundles and filtering connections deemed harmless. Another concern was the close proximity of the data points along the outer circle. Users felt it made it difficult to click on the points when using features such as drill down. One approach to alleviating this problem is to implement a magnification lens on the cursor, which would expand the areas of the visualization over which the cursor is hovering.

5.2.3 NetBytes Viewer

Some analysts had reservations about the NetBytes Viewer being 3D, suggesting it would be better to use the Activity Viewer along with different colours to represent volume ranges. Others liked the fact that the 3D graph could rotate and that the selection mode with the 2D graphs helped to eliminate occlusion. Other analysts wanted more control over choosing which attributes were applied to which data dimension. One analyst suggested that instead of using time on one dimension, one could use source port versus destination port versus volumes. Flexibility and reuse were identified as key requirements.

Analysts also mentioned that the colour of the data points in the 2D graphs should match the colour that the points were given in the 3D graph (currently all data points in the 2D graph are pink). They thought that this might provide better context for the user as he is switching between visualizations. Such a feature would work well for the Time versus Volume graphs as data along that axis is one colour per port/protocol. However, data points along the Port/Protocol versus Volume axis are several different colours. As a result, such a feature could be difficult to implement on that particular axis because the application would have to keep track of the colour of each spike.

Another feature that analysts requested is the ability to compare data between two time periods. For instance, they may want to compare the volume traffic between this month and last month on the impulse graph by overlaying the data to see how patterns have changed over time. The ability to do historical comparisons is extremely important to the analyst; therefore, such a feature is being considered.

Chapter 6

Conclusion

Security visualization helps security analysts conceptualize large network data sets using the high-bandwidth capabilities of human vision. In this thesis, the goal is to build a security visualization system that provides insight for the user when monitoring her network. To provide this insight, it was necessary to tackle some of the key issues facing current security visualizations. These issues pertain to data occlusion and information overload associated with displaying millions of data points on the screen at once. The approach was to limit the number of data points on the screen; thus, showing an overview of the network at a high level and then drilling down on portions of that network through the use of multiple visualizations to give the user several different views. As a result, the FloVis framework was developed — an extensible platform for the creation and integration of security visualizations into a single application. FloVis is designed to contain a suite of interactive visualizations to show several views of network data in support of network security analysis. Since the framework is extendable, analysts can mix and match visualizations that they find most useful and researchers can develop new visualizations. Furthermore, the framework supports a custom event manager that allows visualizations to launch one another using common data parameters. If one visualization does not show a specific pattern another one might. The framework itself allows for seamless transitions between visualizations which enables the user to get more detail by launching one visualization to another.

Next, a set of example visualizations were created for the framework: Activity Viewer [40] (not described in this thesis), FlowBundle and NetBytes Viewer. FlowBundle limits the impact of data occlusion by applying edge bundling [17], drill down, distortion [25], and node aggregation. It is a connection-oriented graph designed to show the relationship between data attribute pairs (*e.g.*, source IP, destination IP, source port, destination port). NetBytes Viewer was created as a complement to

FlowBundle, allowing for a more detailed drill down to an individual host. It shows traffic volume information of individual entities (hosts or subnets) on a network categorized by port or protocol. Furthermore, NetBytes provides a third dimension (time), which gives a historical context to the data so that analysts can detect changes in patterns without having to rely on short term memory. Lastly, the SiLK Query Tool is a front-end to the SiLK toolkit, which allows the user to access the raw SiLK data — the lowest level of drill down possible. The suite of visualizations is designed to work with the SiLK network analysis toolkit. SiLK processes raw NetFlow records into bags and sets, which provide the basis for FloVis views. Data is stored in intermediate data stores to provide a separation between SiLK and the framework so that other data processing toolkits are possible.

The approach was evaluated using two methods. First, three different case studies of suspected malicious behaviour were developed. They showed how FloVis can be used to identify and analyze the behaviour. Next, an informal user study was conducted with a dozen security analysts. The overall feedback was positive as the analysts found value in our approach. They also offered suggestions for improvement in future versions.

The main contribution of the FloVis framework is the development of a platform which is customizable. Not only does FloVis allow security analysts to create an environment that meets their needs, but also, it provides a means of integration for applications built by security visualization researchers. Researchers can focus on building their own arbitrary visualizations and then use FloVis to integrate them and create a fully functional visualization system. Furthermore, the framework reduces application development time so that researchers can concentrate on research rather than development.

Another key contribution of this work is to apply new visualization techniques to the security domain in order to deal with some of the issues facing current visualizations. These issues include:

- Data occlusion occurs by trying to display too many data points on the screen at once.

- Lack of data point labeling due to points being too close together.

- Lack of interactive features including drill down and zooming.

In order to deal with these issues, we used node aggregation to limit the number of data points along with connection bundling to eliminate occlusion.

6.1 Future Work

Although FloVis development has already enjoyed a positive reception, there is still plenty of scope for future work. This section describes a number of future directions for the framework. It is divided into three sections: additions to the framework, new features for existing visualizations, and new visualizations.

6.1.1 Additions to the Framework

One of the major features to be added to the framework is view annotation capabilities. This would involve adding a set of drawing functions (*e.g.*, lines, arrows, circles) such that the user could mark interesting patterns in a visualization and subsequently save the visualization for future reference. Another component of this might be to integrate FloVis with a web-based knowledge management system whereby the user could create simple notes and reports on findings and then save the notes and pictures into a searchable library. This would help support collaboration amongst analysts and also provide a historical data trail for referral when new and interesting patterns arise.

Another direction is the creation of a generic data structure that describes the network data being analyzed. The goal is to design a comprehensive data structure so that one visualization could launch any other visualization in the framework without having to know the details of that visualization. This would facilitate a much more flexible framework and better interaction between plugins. One possibility is to create a data structure that contains data variables similar to the command line parameters provided by `rwfilter`[1]. Each visualization could validate the parameters passed in by another visualization and prompt the user for any missing information.

Next, the creation of a user preferences interface is important so that users can change preferences for different visualizations. For example, one of the security analysts interviewed was colour blind and needs the flexibility to change colour schemes

[1]rwfilter is a NetFlow filtering tool provided by the SiLK Toolkit

to be able to use the tool effectively.

Lastly, another area of investigation would allow multiple plugins to appear on the same tab at once. For instance, FlowBundle might appear on the top part of the tab and the SiLK query tool results appear on the bottom part so the user can look at the raw data along with the visualization. This would likely take some small changes to the plugin interface to accomplish.

6.1.2 New Features for Existing Visualizations

FlowBundle needs a magnifying lens mechanism that would enlarge the portion of the visualization around the mouse cursor. This would allow users to more easily interact with the end points of the connections; furthermore, it would enable them to select certain bundles and connections for better highlighting and filtering capabilities.

Another feature requested is to make FlowBundle not have to rely on contiguous IP address ranges around the outside of the circle. It could allow the user to configure which hosts are assigned to which data points on the visual circle through some sort of configuration file or a filtering capability. This will allow the user to focus on hosts they deem important and remove ranges that are uninteresting.

Other interesting potential features include:

- Adding descriptive icons to data points around the circle. This could include adding a country icon to IP addresses that are from known countries or agency icons for IP addresses from known agencies.
- Experimenting with colour to see how it can be used in other ways (*e.g.*, TCP flags).
- Adding a time dimension to the diagram. One approach might be to use animation (although this relies on the user's short term memory to process).
- Adding drill down capabilities to the bi-directional bundling diagram.
- Experimenting with larger datasets. Access to large datasets is a constant problem.

NetBytes Viewer will also be augmented with bi-directional volume support. In this scenario traffic to a given port will be represented by an upward pulse while traffic from the port will be signified with a downward pulse. It is also desired make the dimensions more flexible such that the user can define the dimensions that he wants to investigate. Finally, analysts requested a mechanism to show volume differences between data from two different time periods in order to see pattern changes over time.

6.1.3 New Visualizations

One of the interesting aspects of analysis discovered when talking to security analysts is that they retain lists of IP addresses. For example, they have lists of known bad hosts, lists of IP addresses that contain potential bad hosts, and lists of known good hosts. As such, investigation of visualization techniques for the organization of IP addresses based on lists, organizations, agencies, departments, networks and subnetworks is needed. The idea is to allow the user to create flexible groups of IP addresses and show how those groups interact with one another at a high level. This type of visualization can be understood as an overview and *Table of Contents* visualization. It will give a high level overview of how the analysts network environment is behaving and interacting (*e.g.*, how much traffic is flowing between groups at any one time.). Furthermore, the visualization would provide the basis for launching other visualizations. For instance, the analyst wants to see if any hosts from the *bad list* have been communicating with hosts from the finance department. He could use the overview visualization to select the bad list and the finance agency and then launch FlowBundle to see host level interactions.

Another interesting visualization to investigate is the *Top N* view. The idea is to create some criteria metrics about the network hosts and then pick the top N hosts that meet that criteria. For instance, the analyst may want to track the top 100 internal hosts that generate the most traffic over a given period and monitor how that list changes over time. New hosts entering the lists could be a sign of malicious behaviour. Such a visualization paradigm would give an analyst a starting point for their analysis. It is intended as a way to draw attention to relatively few hosts in a field of millions.

Finally, there is a need to incorporate new data sources into FloVis visualization paradigms. One such data source is DNS information. Much insight is gained by mapping an IP address to its DNS name. As such, visualizations around clustering IP addresses based on DNS name could be interesting and provide valuable information to the analyst.

Bibliography

[1] Kulsoom Abdullah, Chris Lee, Gregory Conti, John A. Copeland, and John Stasko. IDS Rainstorm: Visualizing IDS Alarms. In *VIZSEC '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, pages 1–10, Washington, DC, USA, 2005.

[2] Aleks Aris and Ben Shneiderman. A Node Aggregation Strategy to Reduce Complexity of Network Visualizaton using Semantic Substrates. *IHCIL Technical Report*, pages 1–8, 2008.

[3] Robert Ball, Glenn A. Fink, and Chris North. Home-centric Visualization of Network Traffic for Security Administration. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and Data Mining for Computer Security*, pages 55–64, Washington DC, USA, 2004.

[4] Brian A. Barsky. A Study of the Parametric Uniform B-spline Curve and Surface Representations. Technical Report UCB/CSD-83-118, EECS Department, University of California, Berkeley, May 1983.

[5] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[6] Sheelagh Carpendale. VisLink: Revealing Relationships Amongst Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007.

[7] Greg Conti. *Security Data Visualization*. No Starch Press, San Francisco, CA, USA, 2007.

[8] Gregory Conti, Mustaque Ahamad, and John Stasko. Attacking Information Visualization System Usability Overloading and Deceiving the Human. In *SOUPS '05: Proceedings of the 2005 Symposium on Usable Privacy and Security*, pages 89–100, New York, NY, USA, 2005. ACM.

[9] Bas Cornelissen, Danny Holten, Andy Zaidman, Leon Moonen, Jarke J. van Wijk, and Arie van Deursen. Understanding Execution Traces Using Massive Sequence and Circular Bundle Views. In *ICPC '07: Proceedings of the 15th IEEE International Conference on Program Comprehension*, pages 49–58, Washington, DC, USA, 2007.

[10] Anita D. D'Amico, John R. Goodall, Daniel R. Tesone, and Jason K. Kopylec. Visual Discovery in Computer Network Defense. *Computer Graphics and Applications, IEEE*, 27(5):20–27, Sept.-Oct. 2007.

[11] Glenn A. Fink, Paul Muessig, and Chris North. Visual Correlation of Host Processes and Network Traffic. In *VIZSEC '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, pages 11–19, Washington, DC, USA, 2005.

[12] Fabian Fischer, Florian Mansmann, Daniel A. Keim, Stephan Pietzko, and Marcel Waldvogel. Large-scale Network Monitoring for Visual Analysis of Attacks. In *VizSec 2008: Proceedings of the Workshop on Visualization for Computer Security*, pages 111–118. Springer Berlin / Heidelberg, 2008.

[13] George W. Furnas. Generalized Fisheye Views. *SIGCHI Bulletin*, 17(4):16–23, 1986.

[14] John R. Goodall. User Requirements and Design of a Visualization for Intrusion Detection Analysis. *Proceedings from the Sixth Annual IEEE Information Assurance Workshop*, pages 394–401, June 2005.

[15] John R. Goodall, Wayne G. Lutters, Penny Rheingans, and Anita Komlodi. Focusing on Context in Network Traffic Analysis. *IEEE Computer Graphics Applications*, 26(2):72–80, 2006.

[16] Yusuke Hideshima and Hideki Koike. STARMINE: A Visualization System for Cyber Attacks. In Kazuo Misue, Kozo Sugiyama, and Jiro Tanaka, editors, *APVIS*, volume 60 of *CRPIT*, pages 131–138. Australian Computer Society, 2006.

[17] Danny Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[18] Cisco Systems Inc. Introduction to Cisco IOS NetFlow. On Cisco Systems website, Oct 2007.

[19] Barry Irwin and Nick Pilkington. High Level Internet Scale Traffic Visualization Using Hilbert Curve Mapping. In *VizSec 2007: Proceedings of the Workshop on Visualization for Computer Security*, pages 147–158. Springer Berlin / Heidelberg, 2007.

[20] Brian Johnson and Ben Shneiderman. Tree-maps: a Space-Filling Approach to the Visualization of Hierarchical Information Structures. In *VIS '91: Proceedings of the 2nd Conference on Visualization*, pages 284–291, San Diego, California, 1991.

[21] Daniel A. Keim, Florian Mansmann, Jörn Schneidewind, and Tobias Schreck. Monitoring Network Traffic with Radial Traffic Analyzer. In *Visual Analytics Science And Technology, 2006 IEEE Symposium*, pages 123–128, November 2006.

[22] Anita Komlodi, Penny Rheingans, Utkarsha Ayachit, John R. Goodall, and Amit Joshi. A User-centered Look at Glyph-based Security Visualization. In *VIZSEC '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, pages 21–28, Washington, DC, USA, 2005.

[23] Kiran Lakkaraju, William Yurcik, and Adam J. Lee. NVisionIP: Netflow Visualizations of System State for Security Situational Awareness. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 65–72, Washington DC, USA, 2004.

[24] Stephen Lau. The Spinning Cube of Potential Doom. *Communications of the ACM*, 47(6):25–26, 2004.

[25] Ying K. Leung and Mark D. Apperley. A Review and Taxonomy of Distortion-oriented Presentation Techniques. *ACM Transactions on Computer-Human Interactions*, 1(2):126–160, 1994.

[26] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The Perspective Wall: Detail and Context Smoothly Integrated. In *CHI '91: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 173–176, New Orleans, Louisiana, United States, 1991.

[27] Florian Mansmann, Daniel A. Keim, Stephen C. North, Brian Rexroad, and Daniel Sheleheda. Visual Analysis of Network Traffic for Resource Planning, Interactive Monitoring, and Interpretation of Security Threats. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1105–1112, 2007.

[28] John McHugh. Sets, Bags, and Rock and Roll: Analyzing Large Data Sets of Network Data. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *Proceedings of the 9th European Symposium on Research in Computer Security*, volume 3193 of *Lecture Notes in Computer Science*, pages 407–422, Sophia Antipolis, France, September 2004. Springer.

[29] John McHugh, Carrie Gates, and Damon Becknel. Situational Awareness and Network Traffic Analysis. In *Proceedings of the Gdansk NATO Workshop on Cyberspace Security and Defence: Research Issues*, volume 196 of *NATO Science Series II. Mathematics, Physics, and Chemistry*, pages 209 – 228, Gdansk, Poland, September 2004.

[30] Jonathan McPherson, Kwan-Liu Ma, Paul Krystosk, Tony Bartoletti, and Marvin Christensen. PortVis: A Tool for Port-based Detection of Security Events. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 73–81, Washington DC, USA, 2004.

[31] John Oberheide, Michael Goff, and Manish Karir. Flamingo: Visualizing Internet Traffic. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium*, pages 150–161, 2006.

[32] Torben Bach Pedersen and Christian S. Jensen. Multidimensional Database Technology. *IEEE Computer*, 34(12):40–46, 2001.

[33] Doantam Phan, John Gerth, Marcia Lee, Andreas Paepcke, and Terry Winograd. Visual Analysis of Network Flow Data with Timelines and Event Plots. In *VizSec 2007: Proceedings of the Workshop on Visualization for Computer Security*, pages 85–99. Springer Berlin / Heidelberg, 2007.

[34] Ramana Rao and Stuart K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. In *CHI '94: Conference companion on Human factors in computing systems*, page 222, Boston, Massachusetts, United States, 1994.

[35] Ronald A. Rensink, J. Kevin O'Regan, and James J. Clark. To See or Not to See: The Need for Attention to Perceive Changes in Scenes. *Psychological Science*, 8:368–373, 1997.

[36] George G. Robertson and Jock D. Mackinlay. The Document Lens. In *UIST '93: Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, pages 101–108, Atlanta, Georgia, United States, 1993.

[37] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society.

[38] Teryl Taylor, Stephen Brooks, and John McHugh. NetBytes Viewer: An Entity-based Netflow Visualization Utility for Identifying Intrusive Behavior. In *VizSEC 2007: Proceedings of the 2007 Workshop on Visualization for Computer Security*, pages 101–114, Sacramento, USA, 2008.

[39] Teryl Taylor, Diana Paterson, Joel Glanfield, Stephen Brooks, Carrie Gates, and John McHugh. FloVis: A Netflow Visualization Tool Abstract. Scottsdale, Arizona, United States, January 2009. *FloCon 2009*. A pdf of the slides is available from the FloCon 2009 web site, `http://www.cert.org/flocon/2009/proceedings.html`.

[40] Teryl Taylor, Diana Paterson, Joel Glanfield, Stephen Brooks, Carrie Gates, and John McHugh. FloVis: Flow Visualization System. In *Proceedings of the 2009 Conference on Cybersecurity Applications and Technology for Homeland Security*, Washington, DC, USA, March 2009.

[41] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[42] Xiaoxin Yin, William Yurcik, Michael Treaster, Yifan Li, and Kiran Lakkaraju. VisFlowConnect: Netflow Visualizations of Link Relationships for Security Situational Awareness. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM*

Workshop on Visualization and Data Mining for Computer Security, pages 26–34, Washington DC, USA, 2004.