SoftSpot : Visual Inspection of Network Intrusion Data

Dustin Lang
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada V6T-1Z4
dalang@cs.ubc.ca

1. Introduction

SoftSpot is a tool for the visualisation of computer network intrusion data. The name *SoftSpot* is derived from the fact that it is a *soft*ware tool that allows the user to *spot* security holes (*soft spots*) in a network.

In security-sensitive computer networks, it is not uncommon to log information about each packet traversing the network. In the event that a computer on the network is compromised ('cracked'), is it crucial that the security team quickly determine what weakness was exploited, how, and by whom. The logged packet data often contains enough information to determine the nature of the attack, but is in a form that is difficult to make sense of, even for knowledgeable network administrators. There is a large volume of data, much of which is irrelevant.

The data consists of text log files produced by a packet sniffer such as `tcpdump`. Each line contains information about one packet traversing the network, and contains fields describing the source and destination machines and ports, a timestamp, connection identifiers, and several flags.

The task is to determine which packets caused the system to be 'cracked'. Once this has been determined, the identifying properties of these packets must be determined so that the 'attack signature' can be described. It would also be useful to determine the identity of the intruder. Further, if the history of the attack can be traced, other areas of weakness of the network may be identified. For example, if the intruder carried out a careful information-gathering phase before launching a pinpoint attack, this could prompt an audit of the areas of the system that were used to gather information, even though these were not directly compromised by the attack.

Once the attack signature has been determined, this information can be added to an existing Network Intrusion Detection System (NIDS). The NIDS will then be able to protect the network from this type of attack in the future.

This is suitable task for information visualisation for several reasons. First, human analysis is required. Automated intrusion detection systems can easily prevent known types of attacks, but new attack types must be analyzed to determine their attack signatures. Second, visualisation may be useful. The data sets are large, and most of the packets are benign. The raw data is cryptic; detecting patterns by inspection of the text files would be challenging. When presented visually, patterns should be more easily discernible. Finally, rapid analysis is important. Since the network remains vulnerable until the attack signature has been determined, it is crucial that the analysis be conducted quickly. Therefore, a tool that speeds the analysis process could be very useful.

2. Related Work

There has been a large amount of work in the visualisation of networks. However, network intrusion data is quite different; the topology of the network is largely irrelevant to the task. Rather, it is the packets on the network that are important. Since a network packet sniffer can only 'smell' packets on the local network segment, the raw packet data has no hierarchical or graphical structure. Therefore, work aimed at visualising the Internet and other computer networks is of little use for intrusion data visualisation.

Several researchers have approached the problem of presenting packet data visually. *VisuSniff* [5], for example, is designed as a tool to help students learn networking concepts. As such, it is designed to deal with a fairly small number of hosts and packets. Packets are grouped into connections, and the hosts and connections are shown in an overview. Selecting a connection shows a sequence diagram of the packets sent between the hosts, and selecting a packet shows the details for that packet. The *Visual TCP/UDP Animator* [6], similarly, is designed for educational purposes and uses a layered approach. While this layered approach could be useful, the visualisations used by *VisuSniff* and *VTA* would not scale to typical network intrusion datasets, containing thousands of hosts and hundreds of thousands of packets.

Network intrusion data is composed of discrete events in time with several properties. The task of analysing these data sets is largely a matter of filtering. *FilmFinder* and *HomeFinder* [1] are designed to support filtering operations on multidimensional discrete data. In each, a series of sliders is used to select ranges of acceptable values for each dimension. The results are plotted in a scatterplot. In both systems, the axes remain fixed. In *HomeFinder*, the scatterplot shows geographical location. Given that location is an important property of homes, this seems reasonable. In *FilmFinder*, however, the axes are also fixed, apparently arbitrarily, to be popularity and year of release.

3. Design

A design for *SoftSpot* has been developed. At present, not all of the features have been implemented. This section will present the full design, and the following section will discuss the portions that have been implemented.

SoftSpot presents a network intrusion data set as a scatterplot, where each packet is represented as a glyph. A glyph has properties including position, colour, size, and shape. The user is able to select which properties of the packets are mapped to the various glyph properties. The mappings can be changed dynamically.

The basic operation in *SoftSpot* is filtering. A filtering operation can be specified by dragging the mouse in the scatterplot window to select the region of interest. In order to filter based on one or two packet properties, then, the user must map those properties to the axes and select the rectangle of interest. This makes it unnecessary to have a slider bar for each dimension, which conserves screen real estate and makes the display less cluttered. The scatterplot serves as both input and output space.

In order to determine if patterns observed in an attack data set are normal or abnormal, it would be useful to be able to view both the attack data and a 'baseline' data set collected during normal network operation. It should be simple for the user to either display the scatterplots side by side, or switch between the data sets.

In addition to baseline and attack data sets, each filtering operation generates a new data subset. A data set selection window would allow the user to view any of the data sets. The data sets could be represented as a tree, where a filtered set is a child of the original data set. This would provide an intuitive visual representation of the filtering operations that have been performed, and would allow the user to explore several views of the data in parallel. Undoing a filtering mistake would then involve simply switching back to the parent and marking the child for deletion. The history of the filtering operations performed so far would be the path back to the root of the tree.

This filtering interaction method of selecting a rectangle in the scatterplot, and the tree visualisation of filtering operations seem intuitive for a series of conjunctive ('AND') filters, but it is less clear how disjunction ('OR') should be supported. For example, would the disjunction of two filtered sets become a child of one or both sets, or a new top-level set, or would a different visualisation be required?

In addition to filtering of data sets, it should be possible for the user to graphically ‘zoom in’ on a particular region of the scatterplot. This could be achieved with one of several distortion methods, such as those discussed in [3].

Since most of the properties of packets are discrete rather than continuous, it is possible for many glyphs to map to the same point in the scatterplot. It would be useful to have a visual representation of the density of packets in the plot. This could be achieved by shading the background based on glyph density, or the position of each packet could be ‘jittered’ slightly. This way, regions of high density would become clouds of overlapping points. Alternatively, the packet density could be considered to be another property of the packet and could be mapped to one of the glyph properties such as shape or size.

4. Implementation

Network intrusion data sets can be large. For example, the data sets used in the 1997 Information Exploration Shootout [2] comprise twenty minutes of network activity, and contain about 500,000 packets. In order to handle these large data sets, a *MySQL* database [4] is used for back-end data storage. This provides persistent storage, rapid data access, and filtering using a standard query language. The *SoftSpot* software is written in Java.

In order to keep the user interface responsive, it is important that 500,000 data points not be drawn at once. *SoftSpot* samples packets from the database and draws the points as they become available. By first sampling sparsely and filling in more points over time, we are able to display a coarse plot of the data quickly, and fill in details as they become available.

Presently, *SoftSpot* allows the user to dynamically select the mappings from packet properties to glyph properties. This is accomplished with a widget that lists the packet properties on the left and glyph properties on the right. A mapping is created by dragging a vector between source and destination. Redundant codings can be created by dragging several vectors from one packet property to several glyph properties. When a new mapping is created, the data set is replotted. See Figure 4.

A typical *SoftSpot* scatterplot is shown in Figure 4. Time has been mapped to the horizontal axis, and packet number to the vertical. The ranges of the axes are changed dynamically as new data arrives from the database.

At present, selection of a filtering region is implemented but some technical details surrounding the management of several database tables need to be resolved before multiple filtering stages can be performed. The filtering history widget has

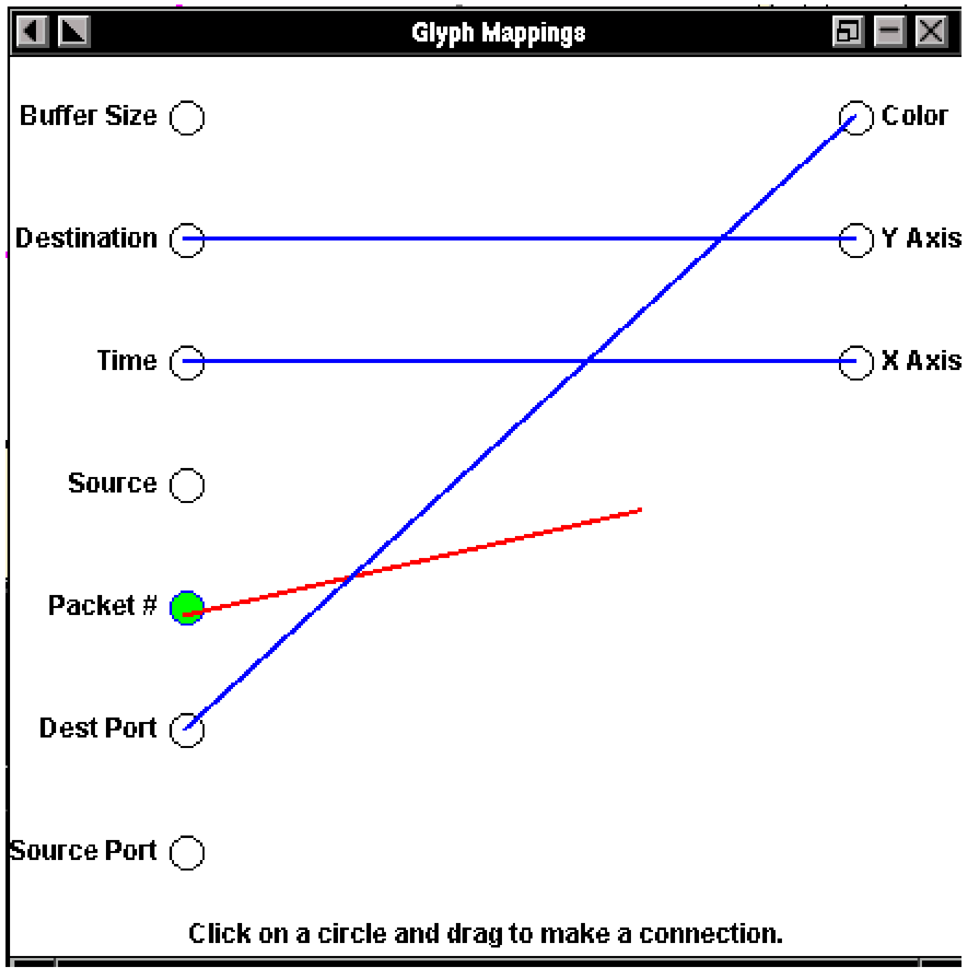

Figure 1: The mapping selection window. A mapping from packet properties to glyph properties is accomplished by dragging a vector from source to destination. The user is currently dragging from “Packet #” field.

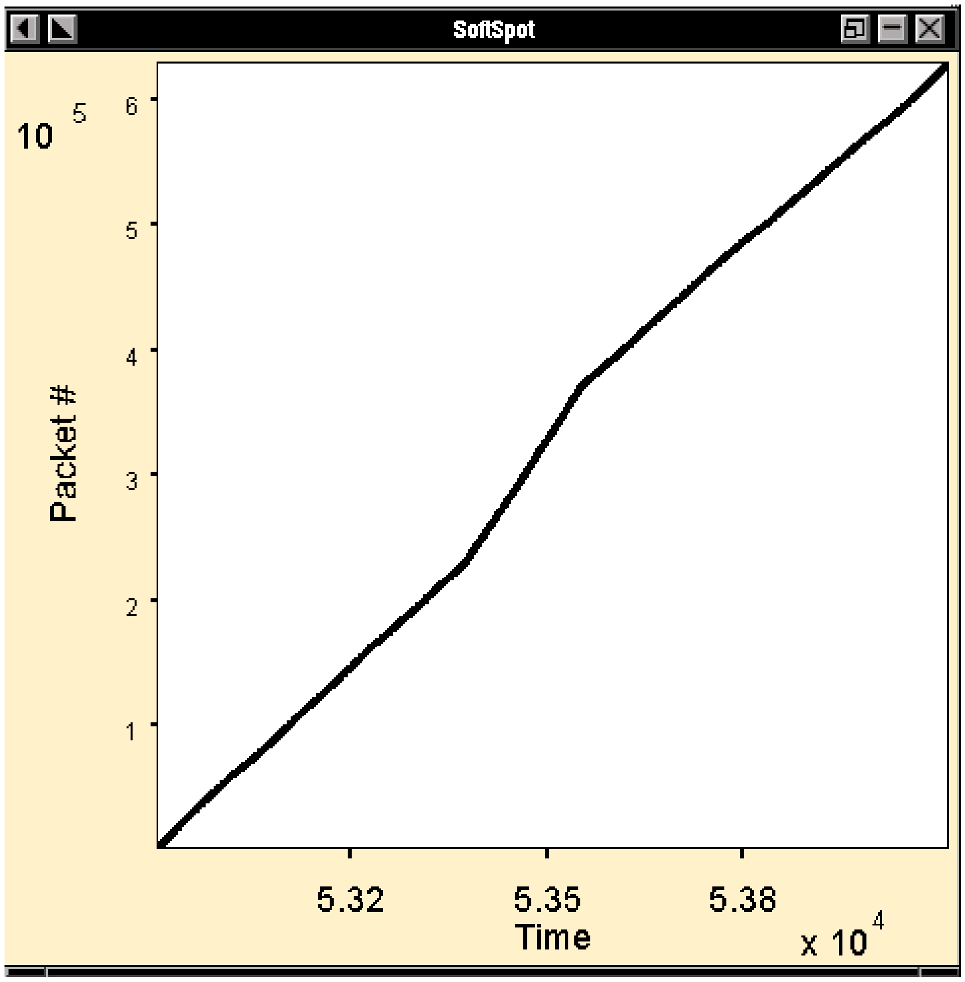

Figure 2: A typical *SoftSpot* scatterplot. This plot contains about 1000 data points and shows packet number against time. There is a clear ‘kink’ in the graph where the slope increases significantly. This corresponds to a burst of network traffic.

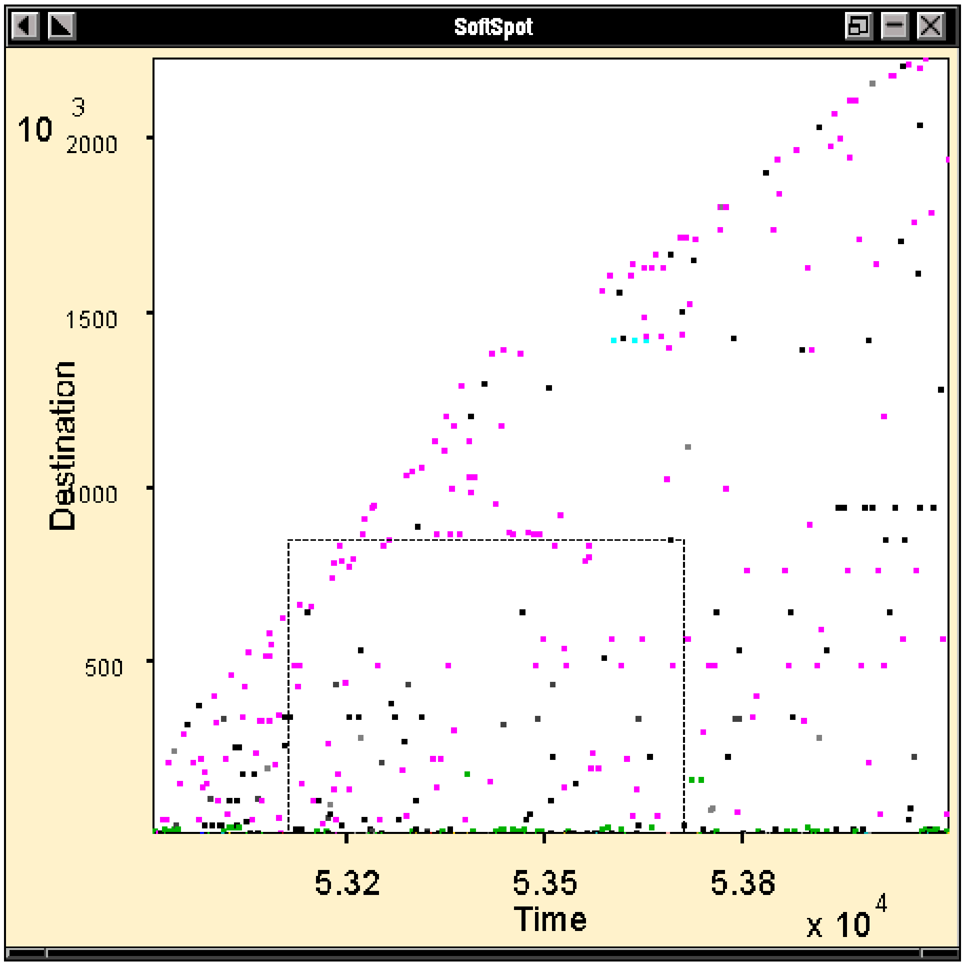

Figure 3: Filtering by selection in *SoftSpot* . The dotted rectangle is the region selected by the user.

also been created, but without filtering properly implemented it is of limited usefulness. Also, focus+context is not currently implemented.

5. Results

As key features of the implementation have not been completed, it is difficult to gauge the effectiveness of *SoftSpot* as a tool for the analysis of network intrusion data. However, the dynamic mapping selection widget seems to be an intuitive method for specifying packet-to-glyph transformations. Likewise, the tree view of multiple filtering operations should provide a simple yet powerful visualisation for a series of filtering operations.

One negative aspect of giving users full control over data-to-glyph mappings is that some mappings will not make as much sense as others. Perhaps the best solution to this problem would be for the mapping widget to suggest good mappings for the selected input by highlighting suitable destinations. This would allow the type of the data to be taken into account while still allowing flexibility.

References

[1] Ahlberg, C. and B. Shneiderman, "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays," *Proceedings of SIGCHI 1994*, pp 313-317.

[2] Institute for Visualization and Perception Research, "Information Exploration Shootout," `http://ivpr.cs.uml.edu/shootout/about.html`

[3] Leung, Y. K. and M. D. Apperley, "A Review and Taxonomy of Distortion-Oriented Presentation Techniques," *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 2, June 1994, pp. 126-160.

[4] MySQL AB. `www.mysql.com`.

[5] Oechsle, R., O. Gronz, and M. Schüler, "VisuSniff: A Tool For The Visualization Of Network Traffic," *Proceedings of the Second Program Visualization Workshop 2002*, pp. 118-124.

[6] Zhao, C. and J. Mayo, "A TCP/UDP Visualization Tool: Visual TCP/UDP Animator (VTA)," *Proceedings of the International Conference on Engineering Education 2002*, pp. 1-6.