Scatter (and other) Plots for Visualizing User Profiling Data and Network Traffic

Tom Goldring
National Security Agency

ABSTRACT

The scatterplot continues to be one of the most useful tools for visualizing numeric data, however what we typically encounter in Computer Security is categorical and/or textual in nature, and how to convert it into a form where scatterplots apply is not always obvious. We outline some simple ideas for doing this and illustrate with two "real data" case studies: User Profiling and Network Traffic. In both cases the results can be quite surprising.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Security

Keywords

Visualization

1. USER PROFILING

1.1 Introduction

Within the context of Computer Security, User Profiling [8, 9, 19, 5, 14, 16, 18, 12, 13, 10, 15] involves two problems: a) *authentication* (is a session being impersonated) and b) *insider misuse* (is the legitimate user doing something they shouldn't). To begin to answer such questions we first need to decide what data from user sessions should be analyzed. Although the majority of published research has focused on Unix command line data, window-based environments are far more ubiquitous in current practice, and in most cases the command line is all but extinct. To focus on user behavior as opposed to system behavior , we collected data from *window titles* (whatever is in the title bar of a window appearing on the desktop) and the *process table* (the mechanism that multitasking operating systems use to keep track of the various applications running concurrently), for the Windows NT operating system. We assume that with very few exceptions, every window appearing on a user's desktop is the result of intentional user activity, and conversely, nearly all user interaction with the system is done through some window. Since windows can be linked to processes via their process ID, it is possible to filter out most if not all background system behavior not consciously initiated by the user. Timing (wall clock and cpu) statistics associated with windows and processes were collected as well. The data set we used is available to the general research community [1] and some experiments previously performed on it are discussed in [7].

VizSEC/DMSEC'04 October 29, 2004, Washington DC, USA

1.2 cpu Usage

Given the availability of timing statistics on a per process level, it is natural to look at how various processes use cpu time, as first proposed in [10]. From the user sessions, we create a list of all process names for windows appearing on the desktop, ordered by total cpu time (over all sessions for all users). We take the top ten (netscape, outlook, winword, ieplore, explorer, msaccess, powerpnt, excel, acrord32, winzip32), call them $P_1, \ldots, P_{10}$ respectively, and let P_{11} represent all the rest. We selected ten users for this experiment, and for $k = 1, \ldots, 10$ created a matrix A_k for user k whose ij'th element is the proportion used by P_j $(j = 1, \ldots, 11)$ of the total cpu time for session i (the sessions are labeled in time order, i.e. session i_1 occured prior to i_2 if $i_1 < i_2$). Then concatenate $A_1, \ldots, A_k$ one underneath the other and add a column on the left containing the user label k for each row. A *data image* plot [11] for this final matrix is shown in Fig. 1. It is derived by normalizing each column to the unit interval ($\min \rightarrow 0,\ \max \rightarrow 1$), associating a rectangular cell with each entry, and coloring the cells according to the normalized value ($0 \rightarrow$ black, $1 \rightarrow$ white). Column 1 clearly shows the user boundaries. Note how for users 2-8, most of their cpu time is dominated by the first five processes. On the other hand, user 1 appears to be somewhat of a maverick, using only explorer (which is the name of many built in processes for the Windows OS) but none of the other top ten. User 8 seems to have spent a number of nearly consecutive sessions making heavy use of acrord32, but hardly ever using it otherwise. msaccess is used in a few sessions by user 2, and almost never by anyone else. These examples illustrate how some users can look very much alike, but at the same time how some of a given user's sessions may appear anomalous in certain ways when compared to that same user's other sessions. Hence the difficulty of the authentication problem for user profiling on this platform.

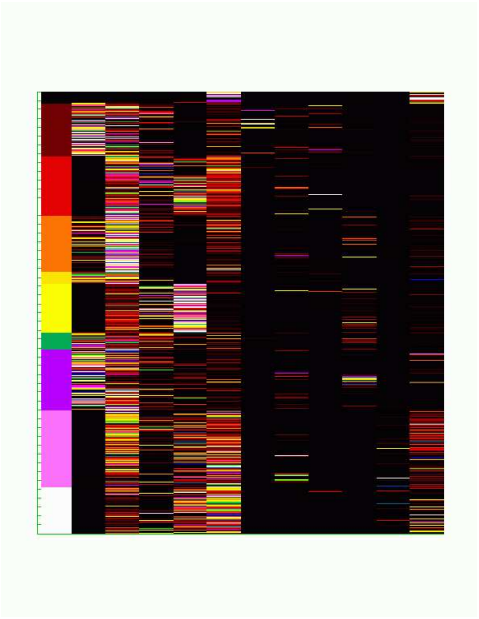

Figure 1: Data Image for Window Processes

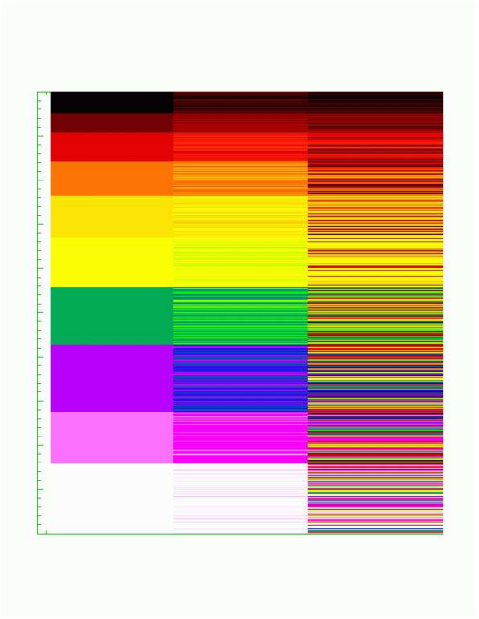

Figure 2: Data Image for Word Repetitions

1.3 Word Repetitions

It turns out that the actual words appearing in the window titles form a reasonably good feature for distinguishing among users [6]. Based on this we would like to visualize to what degree does a given person reuse words from one session to the next, and to what degree are the word sets of two people different. This can be displayed in a scatterplot as follows: take all sessions belonging to the first user and concatenate them, where we restrict the contents of a session to only the window titles, one to a line. Do the same for the other users, and then concatenate all of these into a single text document. So user 1's sessions will appear in lines $1 - L_1$, user 2's sessions in lines $(L_1+1) - L_2$, etc. Now create an index for this document, i.e. each entry is a word appearing in the document and the line numbers that word appears in. We exclude from the index numbers, words "belonging to Windows" (see the documentation in [1]), a given set of stop words, and any words of length less than three. Each word gives rise to a set of triples (u, x, y) where $x > y$ are line numbers and u is the user number for x. So e.g. if "hello" shows up in lines 10, 25, and 36 where $L_1 = 27$, $L_2 > 36$ then we obtain the set $\{(1, 25, 10), (1, 36, 10), (2, 36, 25)\}$.

Since some people generate more and/or larger sessions than others, plots of the derived numeric data can look somewhat uneven. For display purposes, 1000 lines were chosen at random for each user (so $L_n = 1000\,n$ for $n = 1, \ldots, 10$), resulting in 30,551 triples, i.e. a 30551×3 matrix with rows sorted by user number. A data image plot for this matrix is shown in Fig. 2. As before, column 1 shows the user boundaries, and as expected, columns 1 and 2 are strongly correlated. The degree to which columns 1 and 3 are correlated gives an indication of how well the particular words identify the user. For example, if everyone used a common set of words with similar frequency you would expect no correlation at all, whereas if the user's word sets were mutually nonintersecting, the correlation would be very high.

If we now remove column 1 and use it only to color the data points, we obtain the scatterplot of the remaining two columns, shown in Fig. 3. Since $x > y$ for all pairs, the upper half quadrant is empty. The fact that the block corresponding to user 2 ($1001 \leq x \leq 2000$) visually splits into a sparsely filled rectangle beneath a densely filled triangle shows that the word sets for users 1 and 2 are fairly distinct. Similarly for user 3 vs. 1 and 2. The effect for user 4 is slightly less pronounced, while user 5 appears to be better distinguished from users 1, 2, and 4 than from 3. The distinction is very strong for user 6, and quite good for users 7 and 9. User 8 separates from 1 better than from 2–7, while 10 looks very similar to 9 but not to 1–8.

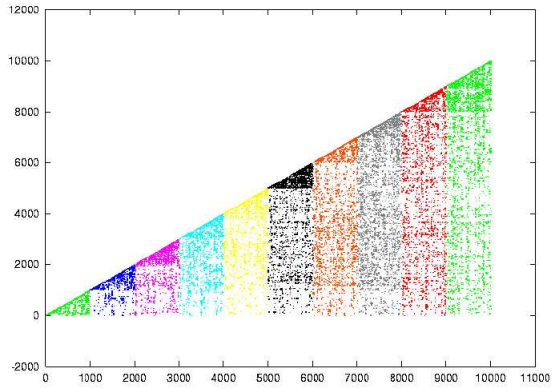

Figure 3: Scatterplot of Word Repetitions

1.4 Interval Graphs

Another way to analyze the user profiling data is through the notion of an *interval graph* [3]. For any session, create a graph whose vertices are the windows opened by the user, with an edge joining two vertices if the corresponding windows are open concurrently. Optionally, the time the two windows are both open may be used as an edge weight. These graphs may be viewed using an environment such as Renoir [4] (see Fig, 4 for a plot of a single session), however

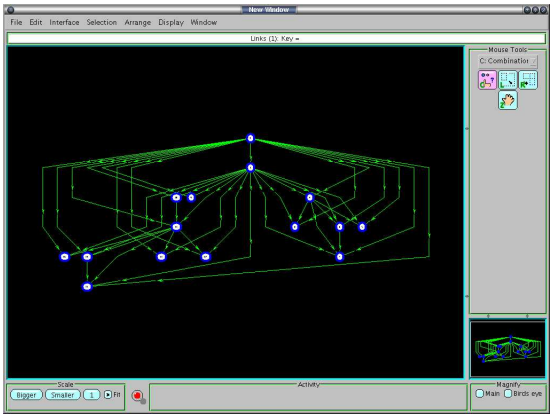

Figure 4: Renoir plot of an Interval Graph

to compare many sessions at once requires a preliminary feature selection step. For example, suppose we choose the following features (attributes) for each interval graph (see any text on graph theory for definitions):

1. number of vertices
2. number of edges
3. length of largest maximal matching
4. number of complete subgraphs
5. number of vertices in largest complete subgraph
6. number of vertices in longest cycle.

In this way we obtain a six element row vector from each user session, then we combine these vectors into a data matrix. To get a scatterplot from the resulting data matrix we need to project onto some two dimensional subspace, e.g. the first two Principal Components. This particular projection maximizes the intercentroid distances among clusters that may exist in the data, and in our experiments we have found it to be extremely useful. For the interval graph features (same user data as before), we obtain Fig. 5. Although these features do not distinguish among users, the structure in the plot, which at present we do not understand, shows that the feature set does provide a very well defined partition for the data. A potential strategy for discovering the meaning of such structure is outlined at the end of the next section.

2. NETWORK TRAFFIC

Some innovative ideas for visualizing computer network traffic are presented in [11], but for the most part, the majority of network traffic visualizations appearing in the literature show (some variation on the theme of) plots of IP

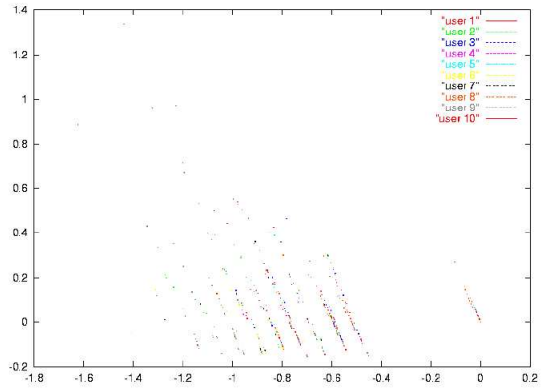

Figure 5: Principal Components of Interval Graph Data

connection graphs, i.e. graphs where the vertices are IP addresses, with an edge between v_i and v_j if a connection exists between the corresponding hosts. While such plots provide a good starting point and in particular do an excellent job of showing things like hubs and port scans, the IP connection graph does not do well at the following:

- showing what happens *inside* a connection
- visualizing more than two or three attributes per node or edge
- displaying many thousands of connections and still having the plot be intelligible
- detecting clusters or patterns arising from interactions among attributes
- employing mathematical decomposition techniques to display the data in various ways.

All of these can be addressed by the lowly scatterplot. As with the interval graphs, we select a set of features from the raw network traffic (either on a per packet level, or on a per session level if the traffic has been sessionized). These can come from packet headers, payloads, or both. We now have a matrix where each row is either a packet or session, and each column contains the value of some attribute. Next, map categorical attributes into numbers, e.g. if an attribute has n possible values $\{a_1, a_2, \ldots, a_n\}$, substitute n columns where $a_1 \Leftrightarrow (1, 0, \ldots, 0)$, $a_2 \Leftrightarrow (0, 1, 0, \ldots, 0)$ etc. (you can reduce the dimension by one using the vertices of an n-simplex). If this adds too many coordinates, another possibility is to use multidimensional scaling to reduce the dimension if a meaningful dissimilarity matrix can be written down. But much more simply, we can just map $a_i \Leftrightarrow i$. This of course introduces an ordering on the attribute values where in fact none may exist, however we do not regard it as a serious problem, because we don't expect it to destroy existing structure. At worst it might introduce additional structure into the scatterplots we are about to describe, but this can be tested by permuting the a_i and redrawing the plot.

We now have a (possibly high dimensional) numeric matrix, and can leverage a host of visualization techniques that exist for multidimensional data such as the data image, parallel coordinates, and tours (e.g. the last two are implemented in the ggobi Data Visualization System [17]). After much experimentation, we have found that projecting onto the first two Principal Components tends to work extremely well in most cases. Some plots obtained in this way from the Lincoln Labs DARPA data [2] are shown below. The results are very intriguing, but at present we don't know what they mean. In theory, this could be addressed through an interactive plotting interface, where the user manually selects and saves a subplot corresponding to outliers or to an interesting pattern (ggobi implements such an interface). From the row numbers, one could then extract the subset of the original data corresponding to the selection and present it to a domain expert for analysis. We are currently implementing this strategy and plan to report results in future work.

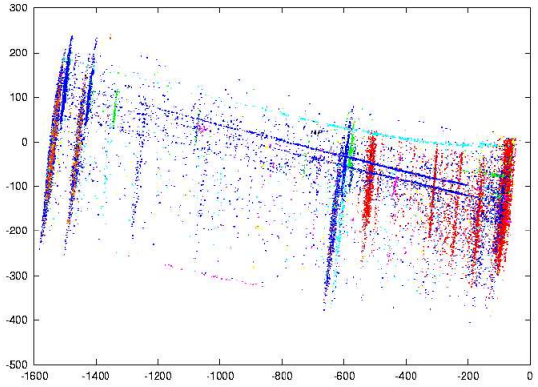

Figure 6: Network Traffic

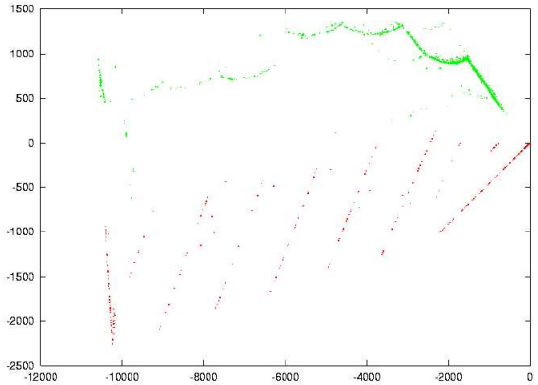

Figure 7: More Network Traffic

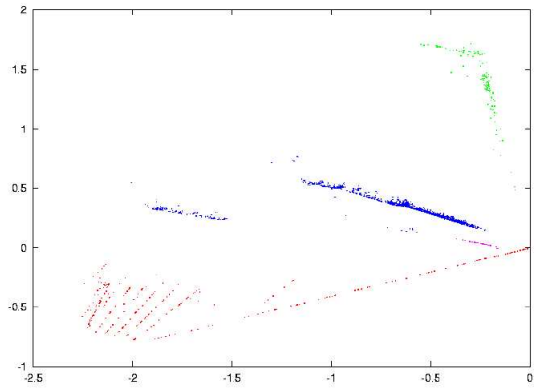

Figure 8: Still More Network Traffic

3. REFERENCES

[1] *ftp://ftp.njit.edu/pub/manikopo/data.*

[2] *www.ll.mit.edu/IST/ideval/data/data_index.html.*

[3] Joint work with m. s. postol.

[4] Renoir: General network visualization and manipulation program. *http://www.nsa.gov/techtrans/techt00013.cfm.*

[5] B. Davison and H. Hirsch. Predicting sequences of user actions. In *AAAI/ICML 1998 Workshop on Predicting the Future*, 1998.

[6] K. DeVault, N. Tucey, and D. Marchette. Analyzing process table and window title data for user identification in a windows environment. *Naval Surface Warfare Center, NSWCDD/TR-03/122*, 2004.

[7] T. Goldring. Authenticating users by profiling behavior. In *ICDM Workshop on Data Mining for Computer Security*. Melbourne, Florida, November 2003.

[8] Javitz and Valdes. The nides statistical component description and justification. *http://www.sdl.sri.com/projects/nides/reports/statreport.ps.gz.*

[9] T. Lane and C. E. Brodley. Sequence matching and learning in anomaly detection for computer security. In *AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*. Providence, RI, 1997.

[10] Y. Liao. Windows nt user profiling with support vector machines. In *Proc. 2002 UC Davis Student Workshop on Computing, Technical Report CSE-2002-28*. Dept. of Computer Science, UC Davis, 2002.

[11] D. Marchette. *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*. Springer Verlag, 2001.

[12] R. Maxion and K. Tan. Anomaly detection in embedded systems. *IEEE Transactions on Computers*, 51(2), February 2002.

[13] R. Maxion and T. Townsend. Masquerade detection using truncated command lines. In *International Conference on Dependable Systems and Networks*. Washington, D.C., June 2002.

[14] P. Reiher. File profiling for insider threats. *Sensors Directorate, Air Force Research Laboratory, AFRL-SN-WP-TR-2002-1102*, 2002.

[15] J. Shavlik, M. Shavlik, and M. Fahland. Evaluating software sensors for actively profiling windows 2000 computer users. In *Fourth International Symposium on Recent Advances in Intrusion Detection*. Davis, CA, 2001.

[16] T. Spyrou and J. Darzentas. Intention modeling: Approximating computer user intentions for detection and prediction of intrusions. In *S. K. Katsikas and D. Gritzalis (eds.), Information Systems Security: Facing the Information Society of the 21'st Century*.

[17] D. Swayne, D. Cook, A. Buja, and D. Lang. ggobi manual. *http://www.ggobi.org/manual.pdf*, 2003.

[18] K. Tan and R. Maxion. Why 6? defining the operational limits of stide, an anomaly-based intrusion detector. In *IEEE Symposium on Security and Privacy*. Oakland, CA, May 2002.

[19] M. Theus and M. Schonlau. Intrusion detection based on structural zeroes. *Statistical Computing & Graphics Newsletter*, 9(1):12–17, 1998.