A Visual Approach for Monitoring Logs

Luc Girardin & Dominique Brodbeck – UBS, Ubilab

ABSTRACT

Analyzing and monitoring logs that portray system, user, and network activity is essential to meet the requirements of high security and optimal resource availability. While most systems now possess satisfactory logging facilities, the tools to monitor and interpret such event logs are still in their infancy.

This paper describes an approach to relieve system and network administrators from manually scanning sequences of log entries. An experimental system based on unsupervised neural networks and spring layouts to automatically classify events contained in logs is explained, and the use of complementary information visualization techniques to visually present and interactively analyze the results is then discussed.

The system we present can be used to analyze past activity as well as to monitor real-time events. We illustrate the system's use for event logs generated by a firewall, however it can be easily coupled to any source of sequential and structured event logs.

Introduction

One of the primary sources of information that enable support for system administration is the logging facilities provided by key system components. Such facilities are often used to track real-time events triggered by system, user, and network activity. Continuously monitoring this activity is of tremendous importance to organizations which rely on high security [Geer, et al., 1997] and optimal resources availability. Logs provide support for proactive system maintenance, anomaly and intrusion detection, failure analysis, and usage assessment.

Conscientious system and network administrators often scan entire log files on a regular basis with little or no tool support. However, the growing use of interconnected computer systems and of the Internet has resulted in a drastic increase of the amount of information contained in such logs, making sequential scans impractical. Existing tools typically utilize rule-based systems to eliminate known-good log entries, and statistical abstractions to simplify the analysis of the activity. Such tools greatly relieve from the repetitive work of scanning log files but generate new difficulties.

The rule creation process forces administrators to draw a clear distinction between what is relevant/irrelevant, severe/benign, critical/isolated, etc... Furthermore, capturing and translating the tacit knowledge of the domain expert into a formal set of rules is difficult and prone to errors. In addition, the resulting rule set usually only covers well-known behaviors, while the most interesting events are often a result of unforeseeable behaviors. Finally, achieving a smooth evolution of the base rule set remains more an art than a science.

The statistical approach, also referred to as data mining, is good at providing abstraction but usually ignores the complexity and the context in which the activity takes place. Moreover, it doesn't support exploratory tasks, such as finding the reason for an increasing number of a certain type of network packets.

Information visualization techniques take advantage of the human perceptual capabilities and provide an overview of the global relationships within a data set while still providing for detailed examination of the underlying data. Examples of visualization techniques for the monitoring of logs may be found in [Couch ,et al., 1996, Hughes, 1996, Karam, 1994].

With machine learning, the goal is to provide automatic classification of the events to permit unattended operations. This approach does usually not rely on any prior knowledge about the content of the data. Applications focussing mainly on computer security, can be found in [Hofmeyr, et al., 1998, Lankewicz, et al., 1997, Tan, 1997].

While both the information visualization and machine learning approaches are promising, no tool combines the potential of both for monitoring event logs. By taking advantage of their strengths, it is possible to provide humans with an interactive display to better understand the activity taking place, while delegating the repetitive tasks to the computer.

Following, we discuss our set of tools, which use a variant of the self-organizing map algorithm and a spring-based layout engine to act as a classifier through multidimensional scaling and topological clustering. To visualize and interact with these representations we make use of the map metaphor, and complementary information visualization techniques such as parallel coordinates and dynamic range sliders to facilitate their investigation.

Visually exploring the log activity provides us with new ways to analyze what is going on and to discover hidden patterns. Our set of components provides

for more effective and user-friendly monitoring of event logs since it focuses on the complexity, letting the computer take care of the simple tasks. Our tools specifically addresses the problems of real-time monitoring and heterogeneous event attributes.

Overview

A logging facility usually stores the activity as a time ordered sequence of events. Logging facilities provide either structured or unstructured descriptions of events. Structured logs contain distinct and uniform entries characterizing each event by a given set of attributes. Most systems, such as firewalls (see Figure 1), accounting systems, and web servers, usually provide us with such structured information.

Unstructured logs do not necessarily contain uniform event entries. For example, the popular syslog facility stores free text entries associated with each event. The individual attributes within such a string may be identified using an indexing technique similarly to the one described in [Chen, et al., 1998] or by judicious parsing.

The attributes form an n-dimensional space, where n is the number of attributes. Each event has a unique position in this space depending on the specific values of its attributes. To put each event into context, we must be able to compare one event to another using a distance or similarity function. Such a function measures the extent to which two elements are related or similar to each other. In our experimental system, we use a modified Euclidean metric. The distance between two characteristics is naively calculated using a simple subtraction for continuous attributes such as time stamps and lengths, and a string comparison for categorical attributes where a match results in a zero, otherwise a one. The overall distance is then obtained by summing all the resulting values. This is subject to a more subjective and empirical definition when domain knowledge is available. For more detail on our metric, please refer to [Brodbeck, et al., 1997].

To provide an overview of this space and portray the activity as a whole, we need to reduce the dimensionality in order to generate a representation of the relative similarities of events, which may be displayed on the screen. The resulting visualization can then serve as a frame of reference to drive further analysis and to embed fine grained tasks such as searching and querying. We choose our low-dimensional space to be a plane, for usability reasons.

Events which are similar will be placed close together while events with unrelated patterns will be further apart. The algorithms we use produce a mapping for each event so that their resulting locations will approximate their similarities and thus form a spatial classification. The resulting representation can be viewed as a map with the particularity of communicating abstract relationships.

We provide two information visualization techniques that support the analysis, filtering, and querying of the information viewed on the map. Parallel coordinates permit the comparison of log entries through visual presentation of their similar and dissimilar properties. Dynamic range sliders are used to highlight a subset of the log entries by interactively specifying the range of each attribute.

Multidimensional Scaling

We use two competing methods to achieve the dimensionality reduction process. Each of the two differs in their output and in their computational complexity. The first uses a method inspired from physics, the spring layout algorithm, while the other is based on an artificial neural network, the self-organizing map algorithm. These two algorithms do not depend on any prior knowledge of the data and in this sense we can say that they exhibit self-organization.

Spring Layout

The spring layout algorithm is based on a physical model where all the data points are mutually connected by springs. The rest distances of these springs are proportional to the respective similarity of the data points in high-dimensional space. The more similar two points are, the shorter the spring between them.

No	Date	Time	Inter.	Origin	Type	Action	Service	Source	Destination	Proto.	Rule	S_Port
2	20Jun98	4:30:01	hme1	pilatus	log	accept	smtp	pilatus-dmz	rigi	tcp	2	42833
3	20Jun98	4:30:01	hme1	pilatus	log	accept	domain-udp	rigi	eiger	udp	1	58942
4	20Jun98	4:30:01	hme1	pilatus	log	accept	domain-udp	rigi	eiger	udp	1	58943
5	20Jun98	4:30:01	hme1	pilatus	log	accept	domain-udp	rigi	eiger	udp	1	58944
6	20Jun98	4:30:01	hme0	pilatus	log	accept	domain-udp	eiger	d.root-servers.net	udp		
7	20Jun98	4:30:01	hme0	pilatus	log	accept	ntp-udp	anapurna	err.ee.ethz.ch	udp		
8	20Jun98	4:30:03	hme1	pilatus	log	accept	domain-udp	rigi	eiger	udp	1	58945
9	20Jun98	4:30:03	hme1	pilatus	log	accept	domain-udp	rigi	eiger	udp	1	58946
10	20Jun98	4:30:03	hme1	pilatus	log	accept	smtp	rigi	eiger	tcp	4	35713
11	20Jun98	4:30:04	hme0	pilatus	log	accept	domain-udp	eiger	ns1.sunrise.ch	udp		
12	20Jun98	4:30:06	hme0	pilatus	log	accept	phdatagram	durhap	192.153.89.255	udp		

Figure 1: The logging information provided by a mainstream firewall. Each event is characterized using a time stamp, the source and destination hosts, the protocol used, whether the connection has been accepted, dropped or rejected, and other relevant information.

The algorithm starts with a random arrangement of the data points in a low-dimensional space. The system is then set free and left to relax with the effect that distant data points which are similar in high-dimensional space are pulled together and close but dissimilar ones are pushed apart. After a number of iterations the system typically stabilizes, resulting in a layout of the data in a low-dimensional space where strongly correlated dimensions are blended together, and topology of the data in high-dimensional space is preserved as best as possible.

The low-dimensional space is defined as three-dimensional at the beginning of the relaxation process and the data points are randomly positioned inside a sphere. We do however introduce a gravity force which slowly pulls the data points towards the ground plane so that we eventually end up with a flat, two-dimensional layout. This process reduces the probability of data points being trapped in a local minimum.

This algorithm may be performed in linear iteration time [Chalmers, 1996]. However, it is still not suitable for datasets containing more than a few thousand entries. For significantly larger datasets, sampling, or pre-clustering using for example the self-organizing map algorithm, can provide a reasonable solution. Parallel computers can also effectively achieve any required scalability by computing the spring forces and the positions concurrently.

The spring layout out algorithm can support real-time processing of dynamic feeds; data points can be gradually inserted at random positions, and older ones removed so as to keep the computational load constant. By shaking the system and letting it run for few iterations, it will quickly stabilize to an appropriate configuration.

Self-organizing Map Algorithm and Categorical Data

The self-organizing map algorithm [Kohonen, 1995] is inspired from biology and imitates two-dimensional maps of the brain made from sensory modalities such as the behavior of the tonotopic map of the auditory region. It is an unsupervised (self-organizing) neural network which generates a feature map that has preserved the topology of the stimuli according to their similarity. The original self-organizing map algorithm is not able to learn categorical values and a variant of the original algorithm has been developed to tackle this limitation.

Self-organizing maps are often used to decrease the number of dimensions while preserving the topological features of a high-dimensional space [Kaski,

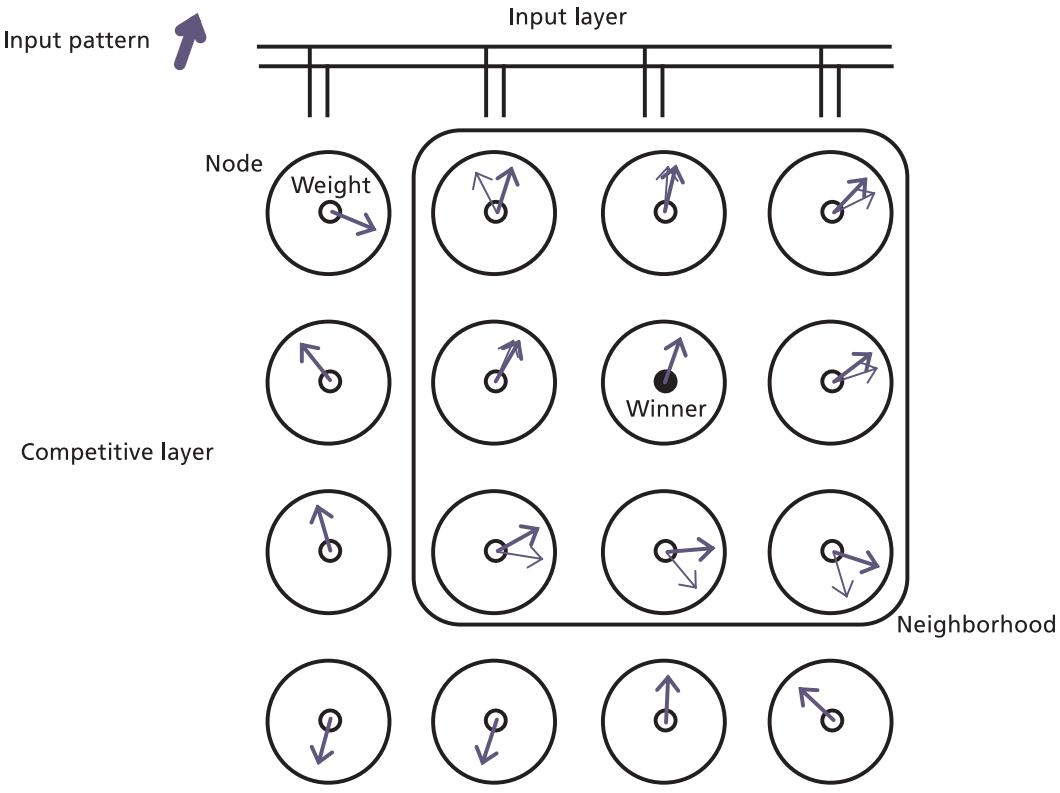

Figure 2: The adaptation process in the self-organizing map algorithm is initiated by feeding the input layer with patterns. Then, the node with a weight that match best the input pattern, the winning node, is sought in the competitive layer. Weights in the neighborhood of the winner can now be slightly adjusted to resemble more closely the input pattern.

1997]. By presenting the feature vectors (in our case the coordinates of each event as a point in an n-dimensional space) to the input layer, the neural network will adapt the weights in the competitive layer to produce a map in the output layer that will exhibit as best as possible the topology of the input space. Please refer to Figure 2 for a description of the adaption process. The nodes in the competitive layer represent a generalization of the possible input patterns.

Our variant algorithm makes use of dynamically growing dictionaries for each weight associated with a categorical dimension. The dictionary contains a set of weighted categories and new categories are dynamically added as needed. Each weight is a vector in a subspace containing only the categories relevant to its associated node. Therefore, the network can learn categorical attributes without exhausting the computing and storage capabilities of the machine.

The self-organizing map algorithm outperforms other methods [Li, et al., 1995], the most popular technique being principal component analysis (PCA). However, as the major difference between self-organizing maps and metric multidimensional scaling techniques (such as the spring layout algorithm), the self-organizing map algorithm will only try to preserve the order of the distances in the high-dimensional space, and will not perform the more natural metric transformation.

Computing self-organizing maps of static datasets is achieved by randomly training the network with the event log entries, until convergence. The time needed to achieve convergence depends on the complexity of the input space, in contrast to the spring layout algorithm which is constrained by the number of event log entries.

Training the network can be a computing intensive process. The computational complexity can be reduced by lowering the dimensionality of the event logs (reducing the number of attributes) or by reducing the size of the output map (resulting in coarser granularity in the clustering). Alternatively, the algorithm can be easily parallelized at the node level.

To analyze logs in real-time, the network can position new event patterns on the output map. However, the results may become of poor quality if the new patterns do not follow the distribution of the

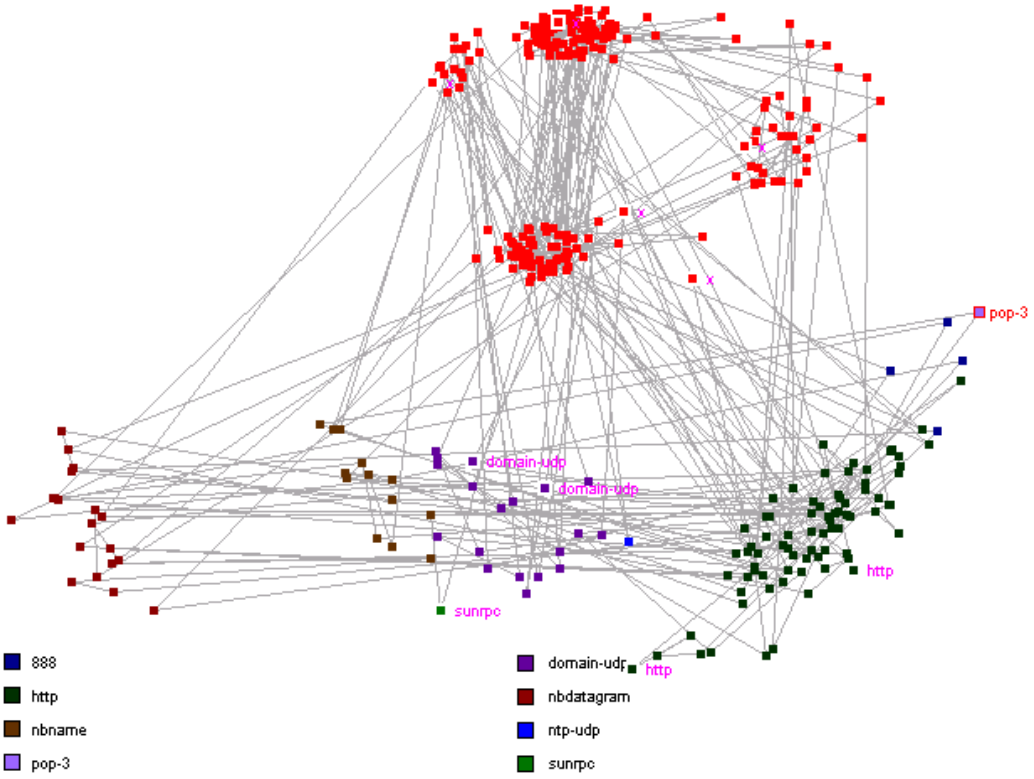

Figure 3: A spring layout of about 500 events from our firewall. Each data point is colored according to the protocol attribute. Data points are sequentialy connected by times, displaying sequences of events with similar characteristics. This visualization clearly establish the similarities and relationships. We can see for example that http requests (in the lower right corner) strongly rely on the domain name service (in the center of the lower part).

events used during the training period. In such a situation, the network may need to be retrained. To avoid the retraining problem, we constantly train the network with new input patterns, and use intermittent noise generation in the weights to insure convergence. However, we are still investigating the theoretical foundation of this strategy.

Visualization

Display of the Spring Layout

To graphically present spring layouts, we use a map metaphor. This provides an overview and establishes a reference system in order to avoid 'getting lost' during the exploration. All logged events are depicted on the map and permit the visual perception of clusters and outliers. Concretely, areas with high density of data points reveal large number of similar activity patterns, while isolated data points correspond to unfamiliar events (see Figure 3). To aid the analysis of the map, some techniques have been used to increase imageability [Brodbeck, et al., 1997]. This approach is ideal for relatively small datasets for which more detailed exploration of the information is desirable. Moreover, maps portraying the activity in real-time have not yet been investigated using this technique and may create usability problems. We currently favor it for analysis of the activity during a limited time frame and for detailed exploration of some nodes within the maps created by the self-organizing map algorithm.

Display of the Self-organizing Map

The self-organizing map results in a discrete space where the units have been topographically ordered. Therefore its display is composed of a fixed number of cells that contain similar events. We developed two visualization techniques to help comprehend the information contained in the network.

The first is to visualize for each cell the characteristics of the last event that it has been assigned. By merging colors, shapes, and textures similarly to the approach of [Levkowitz, 1997], it is possible to code multiple attributes into a single integrated iconic representation (see Figures 4-5). This permits an effective real-time visualization of the changes occurring in the system.

Alternatively, it is possible to use the information contained in the set of weights, which can be seen as a characterization of each node's most typical event. For our purpose, we extract the category with the

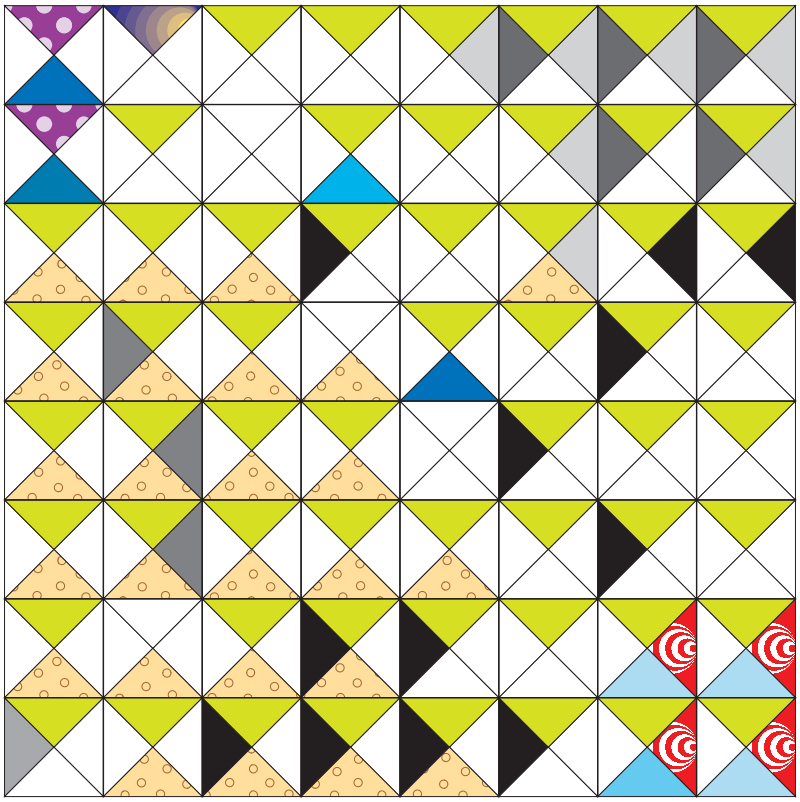

Figure 4: On this display, about 30,000 events have been clustered on a 8 by 8 grid. Each cell depicts some characteristics of the last event which was placed in this location. Four characteristics, which in this case are all categorical attributes, are depicted with triangles using a combination of patterns and colors. The main purpose of this is to provide a visual comparison of cells to determine if they are of similar nature.

dominant weight, which is then compared to the other weights to find the prevalent means of describing a cell. You can find an example in Figure 6. This process is greatly simplified when dealing with numerical values, since the characterization is the weight value itself, and such values may be represented on a continuous scale, for example using simple graphs.

Compared to the spring layout-based maps, the self-organizing map provides for more distinctive identification of clusters and is therefore well suited for larger datasets or real-time monitoring.

Parallel Coordinates

The method of parallel coordinates [Inselberg, 1985] allows the visualization of multidimensional data by a simple two dimensional representation. Through a single view, divergence, trends, and correlations can be analyzed among multiple data points or sets of data points (see Figure 7).

The parallel coordinates system is created by representing each dimension of an n-dimensional space as a vertical axis, spaced apart at a constant distance. A n-dimensional point is then constructed by

Interface	Type	Service	Source	Destination	Proto	Rule
>hme0	accept	nbname	wallace.ubilab.ch	192.153.89.255	udp	17
>hme0	accept	nbname	groenland.ubilab.ch	192.153.89.255	udp	17
>hme0	accept	nbdatagram	mururoa.ubilab.ch	192.153.89.255	udp	17
>hme0	accept	nbdatagram	alice.ubilab.ch	192.153.89.255	udp	17
>hme0	accept	nbdatagram	java.ubilab.ch	192.153.89.255	udp	17
>hme0	accept	nbdatagram	mtblanc	192.153.89.255	udp	17
>hme0	accept	nbdatagram	java.ubilab.ch	192.153.89.255	udp	17
>hme0	accept	nbname	durban	192.153.89.255	udp	17

Figure 5: By selecting a cell in the representation of the self-organizing map, it is possible get the details about each event it contains. In the above example, the cell contains NetBIOS announcements broadcasted on our local network.

tcp	udp	udp	udp	udp	domain-udp	domain-udp	eiger
http	tcp	tcp	udp	udp	domain-udp	domain-udp	domain-udp
http	http	http	tcp	udp	udp	domain-udp	domain-udp
http	http	http	http	udp	udp	udp	udp
america	http	http	http	udp	udp	udp	udp
http	http	http	http	http	udp	udp	udp
http	http	http	http	http	udp	udp	.255
http	http	http	mtblanc	mtblanc	.255	.255	.255

Figure 6: A view of the dominant characteristic of each cell. These values are obtained by scaling each weight over the sum of all weights divided by the number of categories. The present example shows that most of the traffic originates from http requests and domain name queries. Traffic to or from the host ‘mtblanc’ is mainly composed of http requests and broadcasts, while the host ‘america’ seems to be exclusively concerned with http requests. The host ‘eiger’ seems central to domain name queries; it is in fact our primary DNS server.

connection all of its attributes values on their respective axes by a polygonal line.

Dynamic Queries

Dynamic queries [Ahlberg, et al., 1995] are a means to filter event log entries. This technique works by providing a slider for each variable. When one slider is changed, the relevant points are simultaneously updated on the map. Points that fall outside of a slider's range are visually deemphasize on the map using a ghosting effect. One slider is set up for each attribute so that ranges of selected values are AND'd together to form a compound query (see Figure 8).

Practical Use

To investigate the activity taking place in the logs, we can interconnect our visual components to provide for interactive exploration. This allows the users address their multiple needs, using the appropriate tool for a particular task and taking advantage of their possible synergies. In brief, the self-organizing map is used as the global frame of reference, with spring layout maps used for more accurate depiction of interesting areas. Items can be selected on both kinds of map and visually compared using parallel coordinates. In addition, the dynamic range sliders are provided for filtering and querying, using a ghosting effect for all points that don't satisfy the conditions,

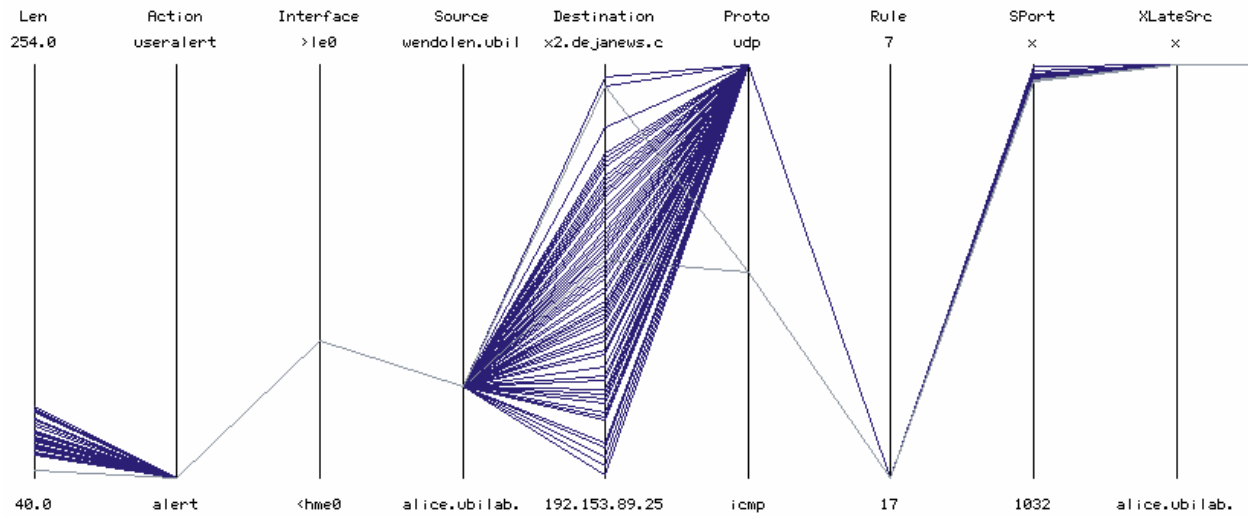

Figure 7: Parallel coordinates visualize multidimensional data points as polygonal lines. In the above example, we compare a cluster with two outliers. We see that they differ in two attributes: they are of shorter length and use the tcp protocol instead of udp.

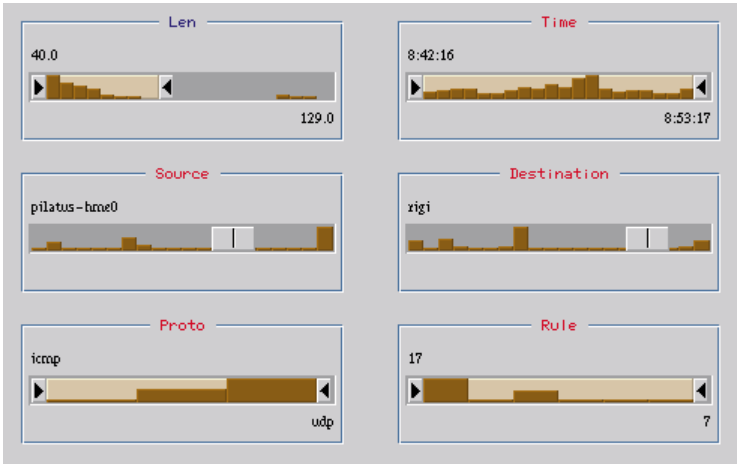

Figure 8: Dynamic queries allow for range selection over each attribute interactively. This simplifies the creation of complex queries. The query above results in the selection of connections of firewall events with short packet lengths, originating from the host 'pilatus-hme0 'and having 'rigi' as the destination. Histograms are integrated into the sliders to show the distribution of data values.

effectively highlighting the rest without loosing context.

By using these components to analyze the connections going through our firewall, we found some interesting applications for our approach. Below we describe a few of the more important ones, and explain how we currently use our tools to perform them.

Usage Assessment

Assessing how the resources are used and how the information flows through the network is important in respect to maximizing the use of the computing and networking infrastructures. During the analyses of our firewall logs, we discovered that two of our master DNS servers were obtaining exactly the same information, inefficiently querying remote sources for information already available locally. This information was obtained by analyzing a few adjacent cells in the self-organizing map using the spring layout. On the resulting map, some events have been laid out in the middle of two clusters containing the domain name activity of our servers. By visualizing them using parallel coordinates, it was obvious that the destination host was similar for both servers. We were able to easily correct this inefficiency and gain more network bandwidth.

Real-time Trend Analyses

By visualizing the evolution of the activity over time, one can create a mental image of the changes occurring in the system. This can be achieved by sporadically getting an overview of the present situation using the self-organizing map, or possibly using a spring layout. This way, we were able to gain a general idea of when people were mainly browsing the web, remote colleagues were sending email to us, and hackers were scanning our network.

Anomaly Detection

It is better to detect and proactively repair problems to maintain a good quality of service, than to wait for users to complain about a service behaving abnormally. Using this tool kit we were visually warned that a network segment was broken because of a high and repetitive number of similar connections.

Break-in Attempts Detection

Intrusion detection deals with the problem of distinguishing between misuse and normal use. It is clear that all types of intrusive behavior cannot be identified in advance [Frank, 1994], and our approach acknowledges this fact. While we didn't manage to discover any real intruder, we have been able to spot different sources scanning our network for security holes, and repetitive trials of abusing some of our services. We expect our approach to be more effective for intrusion detection if used with much more detailed logging information.

Discussion and Future Directions

While developing and continuously evaluating our approach, we were confronted with a number of issues, of which some remain unsolved. Since we want to remain general in our quest to make a better use of logs, we have to date ignored some opportunities to improve the effectiveness of our tools in some specific contexts, especially when domain knowledge is available.

One of the main difficulties we have been faced with is how to most appropriately process and represent events in real-time. To address these issues implies algorithms that can process information very quickly and a user interface that provides the right compromise between dynamics and interactivity. We feel that subsequent research and experience in the area of real-time information visualization is strongly needed.

At the beginning of our research, we decided to experiment with firewall logs since we were suffering from the information overload syndrome. We learned a significant amount while analyzing these logs with our tools, but we now feel that the content of our firewall logs is poor. We would like systems to provide much more detailed descriptions of each event. To cope with this problem, we are now considering the possibility of merging multiple sources of logs to gain more contextual information. At this stage, our approach does not properly make use of the context in which an event takes place. In fact, it considers each event as a separate entity, while taking into account the activity that has taken place before, or eventually after, is certainly of tremendous importance.

In our firewall example, we mainly relied on information containing categorical attributes. There is still too little information design experience to cope with the representation of data that do not fit on a continuous scale or where no particular order makes sense. Our research would greatly benefit from new visualization techniques, especially if they handle sparse categorical datasets.

Currently, help is provided for actively exploring and passively monitoring event logs. However, one would certainly like to automatically carry out actions to affect the activity, such as redirecting overloaded or faulty services to other hosts, disconnecting intruders, or triggering actions by mimicking previous user behaviors. Even if such automatic responses may not have an elegant theoretical fundation, they may nonetheless be of practical interest. In the same spirit, we would like monitoring tasks to be performed collaboratively, but there is currently no integrated tool to support that.

Conclusion

During this work, we have built tools to monitor, explore, and analyse sources of real-time event logs.

Through the use of self-organizing algorithms, the classification of the event logs is performed automatically and does not rely on any apriori knowledge about the content of the logs. We use a map metaphor which provides the user with a frame of reference and the visual tools which enable an intuitive and effective interpretation of the information contained in the map.

Our tool relieves system and network administrators from the laborious task of scanning long log files and building sets of rules for automatic classification. We also expect the users of our tools to discover hidden patterns of activity in their systems.

While the set of components we developed is still in its infancy, we have nonetheless proven that the approach of interweaving machine learning algorithms and information visualization techniques is potentially a powerful approach for anybody confronted with the analysis of activity-based information.

Acknowledgments

We are indebted to a number of people who have contributed in this research. We are especially grateful to Timothy Jones, Vuk Ercegovac, Florian Raemy, Hans-Peter Frei, Kan Zhang, Jan Schultheiss, Matthew Chalmers, Pamela Cotture, Melissa Binde, and Aline Chabloz, for their interesting discussions, corrections, suggestions, developments, and support. Thanks also to our LISA '98 reviewers for their insightful comments and suggestions.

Author Information

Luc Girardin is currently research staff member and system administrator at the UBS IT research department, Ubilab. His research focus on information visualization, collaborative virtual environments, and adaptive and reliable systems. He has worked as researcher, system administrator, or developer for six years, and holds a masters degree in computer science and telecommunications. He can be reached via email at Luc.Girardin@ubs.com.

Dominique Brodbeck received his PhD in Physics in 1992 from the University of Basel, Switzerland. He then joined the IBM Almaden Research Center (San Jose, CA), first as a post-doc and later as a research staff member to work in areas such as scientific visualization and information visualization for data mining. In 1996 he moved back to Switzerland to join Ubilab and now works on visualization of abstract data, information design, information management, and related areas. Dominique can be reached via email at Dominique.Brodbeck@ubs.com.

References

[Ahlberg, et al., 1995] Ahlberg, C.; Wistrand, E. IVEE: ''An Information Visualization & Exploration Environment.'' *Proc. IEEE Information Visualization '95,* Atlanta, Georgia, USA, October 30-31, 1995, pp. 66-73.

[Brodbeck, et al., 1997] Brodbeck, Dominique; Chalmers, Matthew; Lunzer, Aran; Cotture, Pamela. ''Domesticating Bead: Adapting an Information Visualization System to a Financial Institution.'' *Proc. IEEE Information Visualization '97,* Phoenix, Arizona, USA, October 20-21, 1997, pp. 73-80.

[Chalmers, 1996] Chalmers, Matthew. ''A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data.'' *Proc. IEEE Visualization '96,"* San Francisco, California, USA, October 1996.

[Chen, et al, 1998] Chen, Hsinchun; Nunamaker, Jay Jr.; Orwig, Richard, and Titkova, Olga. ''Information Visualization for Collaborative Computing.'' *IEEE Computer*, August 1998, pp. 75-82.

[Couch, et al., 1996] Couch, Alva L. ''Visualizing Huge Tracefiles with Xscal.'' *Proc. 10th Systems Administration Conference (LISA'96)*, Chicago, IL, USA, September 29-October 4, 1996, pp. 51-58

[Frank, 1994] Frank, Jeremy. ''Artificial Intelligence and Intrusion Detection: Current and Future Directions.'' *Proc. 17th National Computer Security Conference,* 1994.

[Geer, et al., 1997] Geer, Dan (editor); Oppenheimer, David L.; Wagner, David A. and Crabb, Michele D. ''System Security: A Management Perspective.'' Berkeley: *The Usenix Association*; 1997. ISBN: 1-880446-85-5

[Hofmeyr, et al., 1998] Hofmeyr, Steven A.; Forrest, Stephanie, and Somayaji, Anil. ''Intrusion detection using sequences of system calls.'' *Journal of Computer Security,* 1998 (In press).

[Hughes, 1996] Hughes, Doug. ''Using Visualization in System and Network Administration.'' *Proc. 10th Systems Administration Conference (LISA'96),* Chicago, IL, USA, September 29-October 4, 1996, pp. 59-66.

[Inselberg, 1985] Inselberg, Alfred. ''The plane with parallel coordinates.'' *The Visual Computer 1*. Springer, 1985, pp 69-91.

[Karam, 1994] Karam, Gerald M. ''Visualization using Timelines.'' *Proc. 1994 International Symposium on Software Testing and Analysis*, Seattle, WA, USA, August 17-19, 1994.

[Kaski, 1997] Kaski, Samuel. ''Data Exploration Using Self-Organizing Maps.'' *Espoo:* Helsinki University of Technology, 1997.

[Kohonen, 1995] Kohonen, Teuvo. *Self-organizing maps.* Berlin; Heidelberg; New-York: Springer, 1995. ISBN: 3-540-58600-8.

[Lankewicz, et al., 1997] Lankewicz, Linda B.; Srikanth, Radhakrishnan, and George, Roy. ''Anomaly Detection using Signal Processing and Neural Nets.'' *Proc. ONDCP International Technology Symposium*, Chicago, USA, 1997.

[Levkowitz, 1997] Levkowitz, Haim. *Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications.* Boston, Dordrecht, London: Kluwer; 1997. ISBN: 0-7923-9928-5.

[Li, et al., 1995] Li, Sofianto; Vel, Olivier de, and Coomans, Danny. ‘‘Comparative analysis of dimensionality reduction methods,’’ *Learning from Data: Artificial Intelligence and Statistics V*, New York: Springer-Verlag, 1995, pp. 323-331.

[Tan, 1997] Tan, Kymie. *The Application Of Neural Networks to UNIX Computer Security*. 1997.