Enhancing Visual Analysis of Network Traffic Using Knowledge Representation

Ling Xiao* John Gerth* Pat Hanrahan†

Stanford University

ABSTRACT

The last decade has seen a rapid growth in both the volume and variety of network traffic, while at the same time, the need to analyze the traffic for quality of service, security, and misuse has become increasingly important. In this paper, we will present a traffic analysis system that couples visual analysis with a declarative knowledge representation based on first order logic. Our system supports multiple iterations of the sense-making loop of analytic reasoning, by allowing users to save their discoveries as they are found and to reuse them in future iterations. We will show how the knowledge base can be used to improve both the visual representations and the basic analytical tasks of filtering and changing level of detail. More fundamentally, the knowledge representation can be used to classify the traffic. We will present the results of applying the system to successfully classify 80% of network traffic from one day in our laboratory.

CR Categories and Subject Descriptors: I.6.9 [Visualization] information visualization, applications of information visualization, visual analytics, H.5.0 [Information Interfaces and Presentation], data management and knowledge representation.

1 INTRODUCTION

The last decade has seen a rapid growth in both the volume and variety of network traffic, while at the same time it is becoming ever more important for analysts to understand network behaviors to provide quality of service, security, and misuse monitoring. To aid analysts in these tasks, researchers have proposed numerous visualization techniques that apply exploratory analysis to network traffic.

The sense-making loop of information visualization is critical for analysis [5]. The loop involves a repeated sequence of hypothesis, experiment, and discovery. However, current visual analysis systems for network traffic do not support sense-making well because they provide no means for analysts to save their discoveries and build upon them. As such, it becomes the analyst's burden to remember and reason about the multitude of patterns observed during visual analysis, which quickly becomes impossible in massive datasets typical of network traffic.

In this paper we present a network traffic visualization system that enables previous visual discoveries to be used in future analysis. The system accomplishes this by allowing the analyst to interactively create logical models of the visual discoveries. The logical models are stored in a knowledge representation and can be reused. The reuse of knowledge creates an analytical cycle as summarized in figure 1. In addition to facilitating the sense-making loop, knowledge representations allow the creation of more insightful visualizations that the analyst can use to discover more complex and subtle patterns.

To evaluate effectiveness, we will present the results of applying our system to analyze one day of network traffic from our laboratory.

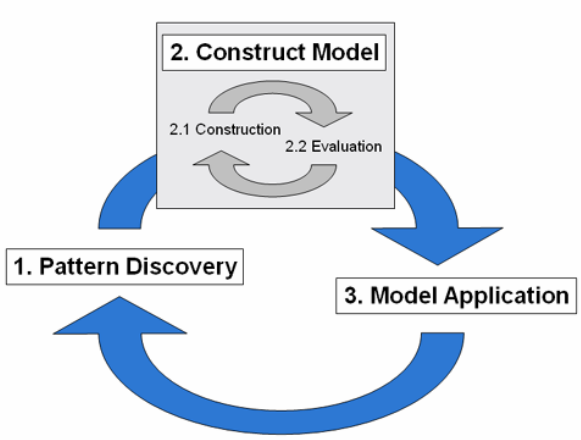


Fig. 1. The knowledge creation process as supported by the system. (1-2) the network analyst creates knowledge by selecting a pattern example on the visualization and interactively constructing and evaluating a model of the selected pattern to input to the knowledge base. (3-4). The system leverages the knowledge base to provide more insightful visual analysis tools, so that analyst can observe and model the behavior of more complex and subtle patterns.

This paper will be structured as follows: section 2 will provide an overview of the visual analysis process; section 3 will give a sampling of related work in this area; section 4 will describe the system's knowledge representation; section 5 will overview the visual knowledge creation; section 6 will demonstrate how the system leverages the knowledge base to improve visual analysis; section 7 will present our results from applying the system; section 8 will discuss the shortcomings of the current implementation; and section 9 will conclude the paper and provide future research directions.

2 SYSTEM OVERVIEW

The system has been designed to leverage the relationship between visual analysis and knowledge. The knowledge base is represented by logical models that describe traffic patterns. The analyst can interactively create logical models representing visual discoveries, and use the knowledge base of logical models to improve future visual analysis. In this section we will provide an overview of this process.

Stage 1: Let us assume that the analyst can create a visualization that shows an interesting pattern – which either conforms to or challenges his domain knowledge. For example, the analyst knows that when a webpage is accessed, multiple

*Email: lingxiao | gerth @graphics.stanford.edu

† Email: hanrahan@cs.stanford.edu

separate HTTP requests are generated. This is reasonable since a web page contains a collection of URLs representing information that needs to be retrieved. Therefore, a temporal visualization showing the traffic of a webpage access should show a sequence of HTTP requests, each of which is a connection to one of the HTTP ports.

Stage 2: The analyst will then attempt to create a logical model that describes the traffic underlying the pattern. To create the logical model of the observed pattern, the analyst selects the data points comprising an instance of the pattern. The system then identifies a collection of predicates from the knowledge base that can be used to describe the selected data. For example, for a webpage access, the analyst selects a single sequence of HTTP requests, causing the following predicates to be identified: "from same machine", "to same machine", "high temporal locality", "high source port locality", "destination port HTTP".

The analyst then interactively creates a logical clause from these predicates. This clause is iteratively constructed (stage 2.1) and evaluated (stage 2.2). For example, the analyst may first select the predicate "to same machine", but find that it is too general, and therefore add other predicates such as "high temporal locality", etc. to the conjunction, until he creates a clause that models the web page access pattern correctly, which may be: "from same machine" and "to same machine" and "high temporal locality" and "destination port HTTP".

Stage 3: Once the analyst has created a logical clause describing the pattern, the system incorporates the model into the knowledge base. The model can then be applied to all the data in the system to label the traffic of that type. For example, the system will label all the flows that are accepted by a web page access model with the "web" label.

The analyst can now leverage the augmented data to create more insightful visualizations, and perform further visual analysis. For example, the analyst can now visualize only web traffic (all the traffic that satisfies the web page load model), or he can filter out the web traffic and concentrate his efforts on the remaining traffic.

Using the enhanced visualizations may then lead to other patterns being discovered (stage 1), modeled (stage 2) and incorporated (stage 3). Thus, the analyst uses the system to iteratively build upon his prior visual discoveries.

3 RELATED WORK

Visualizations of internet network traffic patterns have existed for over a decade [4, 14]. Early visualizations were aimed at understanding overall network loads and topology. These have evolved into tools for monitoring quality of service at the scale of the major network trunks [11]. More recently a host of visualizations have been developed in both the research [7, 3, 9] and commercial worlds [1, 15] to help analysts understand traffic at smaller scales ranging from the corporate intranet down to individual machines not only for quality of service but also for network security. Because the volume of network traffic data is so large, most visualization systems provide aggregation and/or filtering mechanisms to reduce clutter and help the analyst focus on the traffic of interest. This is typically achieved by employing standard interactive dialog boxes, range sliders, dynamic queries, brushing, etc.

Our work is most closely related to that found in NVision IP [18] which has recently added monitoring of user actions [12]. In NVision IP, the analyst's selection actions are recorded as a pattern tree using a rule set based on the well-known Berkeley Packet Filter language [13]. These patterns may be saved in a file and subsequently applied to other network datasets, thereby permitting the analyst to capture the value of a particular set of filtering operations as reusable knowledge. We expand the notion of reuse by using first-order logic to capture the analyst's domain expertise in a knowledge base which goes beyond flow attributes, and is extended as he leverages the knowledge base to perform more effective visual analysis.

4 KNOWLEDGE REPRESENTATION

In this section we will describe the type of network traffic that the system can analyze and the knowledge representation that is currently used.

4.1 Data

Currently the system is designed for the analysis of the network flow data captured from the layer 2 switches by a collector running the open source Argus [2] sensor. Flow data is commonly used in network analysis because it does not contain packet payloads, and is thus more scalable, and avoids the social and legal issues associated with packet content recording.

An Argus flow record contains the following fields:

Dimensions	Measures
GMT Start time	Duration
IP protocol	Source packets
Source IP address	Source bytes
Source port	Source application bytes
Destination IP address	Destination packets
Destination port	Destination bytes
	Destination application bytes

These are augmented by the Source and Destination ASN (autonomous system number – roughly corresponding to the ISP), flags indicating the flow state at time of capture and fields for the local hour and day of week.

4.2 Declarative Language

The declarative language used to model patterns is first-order logic, which is a well studied knowledge representation that is capable of modeling human and machine reasoning [10, 3, 6]. As an example of using first-order logic to describe network traffic, let us reconsider the web page load example from section 2. The clause that describes which HTTP connections form a web page load event is as follows:

```
identical_source_IP(x,y) AND
identical_destination_IP(x,y) AND
time_within_2_seconds(x,y) AND
( destination_port_80(x) AND
  destination_port_80(y))
```

The variables x and y represent individual flow records. This logical clause represents the knowledge that a web page load consists of two flow records to port 80 (associated with HTTP traffic) within 2 seconds between the same pair of machines. Since parameters are not part of the universe of discourse, it is necessary to create multiple predicate for different parameter values. While tedious, we have found that most parameters require only a few values.

Many types of traffic consist of multiple flows. To model knowledge about traffic patterns composed of arbitrarily many flows, we introduce the construct of a variable predicate which is

Table 1. A selection of the types of predicates used in the system. (S) means the predicate involves a single argument; (V) means the predicate involves multiple arguments.

Knowledge Type	Sample Predicate Descriptions
Location	(S) Dest IP is DNS server (S) Src ASN is Google
Connection	(S) Protocol is TCP (S) Dest port is 80
Traffic characteristics	(S) Total bytes sent is > 3000 (S) The duration is < 1 s
Temporal	(V) One event before another (V) In tight time sequence
Identical	(V) Has same src IP (V) Has same dest AS number
Counts	(V) Number of arguments
Approximate	(S) Data is within 2KB of 20KB
Order	(V) First argument has dest port 21 (V) Last argument has dest port 514
Trend	(V) Amount of data is increasing (V) Dest IP number is increasing
Variability	(V) High distinct dest port usage (V) High dest IP access rate

a predicate that can accept any number of arguments. The following clause generalizes the previous pattern describing a web page load to any number of HTTP connections:

identical_source_IP(x1,x2,…) AND
identical_destination_IP(x1,x2,…) AND
time_sequence_2_seconds(x1,x2,…) AND
(destination_port_80(x1) AND
  destination_port_80(x2) AND …)

To model network traffic, we initialize the system with the built-in predicates in table 1. These predicates are organized by the type of knowledge that is represented. Some represent knowledge about the source or destination, the type of connection, the characteristics of the traffic, temporal relations and trends, variability etc. Others are provided for efficiency. The system also provides an interface for analysts to add custom predicates.

4.3 Applying Models

Each model in the knowledge base is stored as a label definition pair <*Label*, *Clause*>, where the *Label* is the name for the pattern, and the *Clause* is a logical expression representing the pattern.

To apply a label definition to the flow events, the system produces all the true groundings of the clause. A true grounding is a set of events for which the clause evaluates to true. The label is then associated with each element of a true grounding.

If the clause contained only 1 variable, then the system will append a column with the label name to any table that contains an event satisfying the clause. The label thus becomes an additional attribute of the object and can be used accordingly.

If the clause grounding contains more than 1 event, the system will append the label to each event that is a member of a true grounding as above. In addition, the system will also create a collection of derived events, each of which corresponds to a true grounding of the clause. For example, if the set of flows {a,b,c} and {d,e} both satisfied the "Web Page Load" label clause, we will create a derived event *abc* that corresponds to {a,b,c}, and another event *de* that corresponds to {d,e}. Intuitively the derived *abc* event represents the web access event that caused the flows {a,b,c} to be observed, and similarly for *de*. The attributes of the derived event are computed (according to user specification) from the attributes of the events in the grounding.

Once created, the derived events become part of the universe of discourse and can be used in predicate arguments, and can have other derived events built from them. Hence, derived events permit the analyst to reason at multiple levels, from the flow events at the OSI transport level, up to derived events at the application level and beyond.

5 Visual Knowledge Creation

To create logical models, we have implemented a technique that allows the analyst to select a pattern directly on the visualization, then explore the model space by iteratively constructing and evaluating candidate clauses; eventually converging to a clause that models the pattern according to his domain knowledge. The interactive techniques are only briefly described. Full details can be found in [17].

In this section, we will motivate our approach and buildup the event hierarchy by modeling web crawls based on the web page access events from section 2.

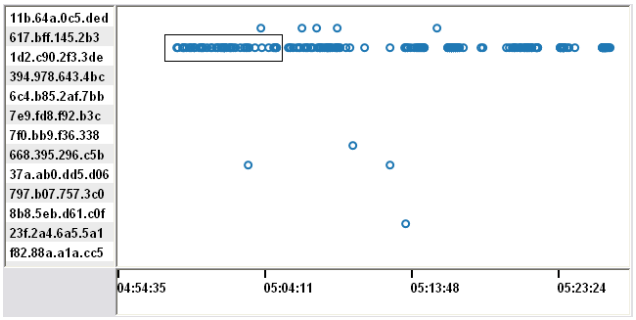
5.1 Visual Representation

To create a model of a pattern, the analyst first needs to observe an instance of the pattern in a visualization, which can be created using domain knowledge, or as part of exploratory analysis. Due to the strong temporal nature of network events, we have found that event diagrams are especially useful. An event diagram is a plot that maps time on the X axis, a dimension or measure on the Y axis, and a circular mark for each event.

In the web crawl scenario, the analyst can use the visualization shown in the figure below, which uses an event diagram to show network traffic from Google to a set of local machines.

5.2 Selecting Patterns

The analyst selects a pattern by selecting the set of marks making



up an instance of the pattern directly on the visualization. Intuitively, the set of selected points is an example of the pattern for which a model is desired.

In the web crawl scenario, the analyst observes that there are rapid bursts of HTTP requests from Google, each of which may be caused by a web crawl. He therefore selects one burst of traffic on the visualization as an example web crawl, as seen above.

5.3 Identifying Predicates

Using the selected data points, the system identifies a list of predicates from the knowledge base that are true for this set of events. These predicates are presented to the analyst as candidate terms for more complex logical clauses.

In the web crawl scenario, the identified predicates include the following:

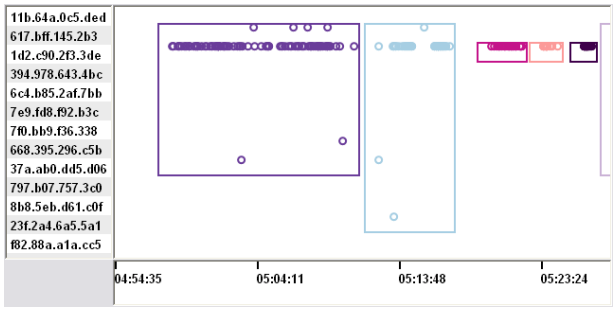
destination_port_80, destination_Stanford,
identical_source_asn, time_sequence_30s,
time_sequence_60s, more_than_4_events,
more_than_32_events

We were quite surprised that the "identical source IP" predicate was not included. This led to the discovery that Google crawls a website using multiple machines, counter to our expectations.

5.4 Constructing a Logical Model

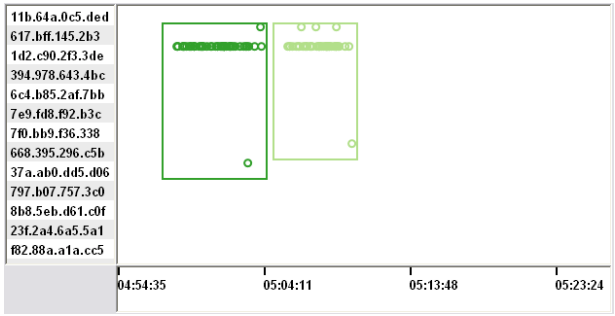
To construct a model, the analyst interactively constructs and evaluates candidate clauses until he finds one that accurately models the desired pattern. To generate a candidate clause, the analyst selects a combination of predicates from the list of predicates. These predicates are formed into a conjunction; that is, the clause identifies those events that satisfy all the predicates. To allow the analyst to evaluate the candidate model, the system immediately shows those groups that satisfy the model. Again, the interactive techniques used are described in [17].

In the web crawl scenario, the analyst knows that a web crawler will access numerous pages possibly from multiple web servers in quick succession. Therefore, he selects the predicates "time sequence 60 seconds" and "more than 4 events". The true groundings of the conjunction are shown below, where each



grounding is a uniform color surrounded by a bounding box.

Unfortunately, this simple conjunction is satisfied by several spurious patterns. In particular, multiple bursts are grouped together (the purple rectangle is really two web crawls), and there seems to be short sequences of HTTP requests that may correspond to human navigation (the small boxes on the right). Therefore, the analyst modifies the model clause by selecting the predicates: "more than 32 events" and "time sequence 30 seconds".



Since this conjunction is more restrictive, only two sets of events satisfy the model. Note that the web crawler accesses several machines (circles at different y locations). This is because web pages frequently reference pages on other web servers.

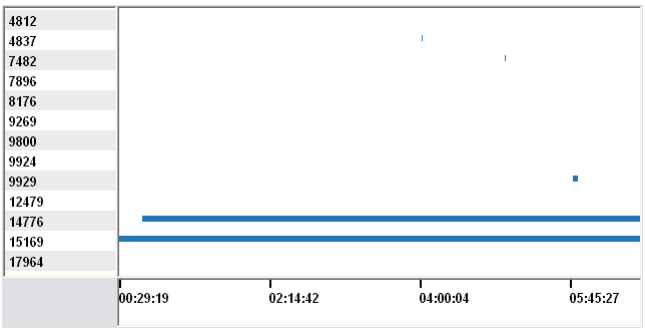
Once the analyst has converged on a clause that models the pattern, he names the pattern, and commits the clause to the knowledge base. In the web crawl scenario, the final clause is:

time_sequence_30s(x1,x2,…) AND
more_than_32_events(x1,x2,…) AND
identical_source_AS_number(x1,x2,…) AND
(is_web_access_event(x1) AND
  is_web_access_event(x2) AND …)

The predicate "identical source AS number" is needed to generalize the web crawl pattern beyond Google.

5.5 Applying the Clause

Once the analyst commits the model, the system applies the model to the entire data set. Below we show the web crawl events from different ASNs. The picture shows that the model successfully generalized to find many web crawls from Google (15169) and also from Inktomi (14776).



6 Using the Knowledge Base

The system is designed to leverage the knowledge base to facilitate more effective visual analysis. In this section, we describe three ways that the system improves visual analysis.

6.1 Controlling Visual Attributes

The knowledge base may be used to control visual attributes of the flows. This results in visualizations that show important knowledge to the analyst, and are hence more insightful. As a simple example, we can map color to the type of traffic associated with the flow as seen in figure 2.

6.2 Changing Level of Detail

Analysts often want to reason at a higher level of abstraction. Given the size of the dataset, looking at low-level flow events often is overwhelming. Changing level of detail is achieved in our system by visualizing derived events.

To demonstrate the use of derived events to emphasize macro patterns in the data, consider the following scenario in which we examine the structure of scan traffic. Figures 3 and 4 show scatterplots of source and destination number of bytes transferred for two types of traffic, fast and slow scans. Figure 3 shows this for individual flow records, while figure 4 shows the aggregated traffic for each individual fast or slow scan. There are much fewer scan events than flow events, reducing the complexity of the display. More importantly, the characteristics of the traffic are

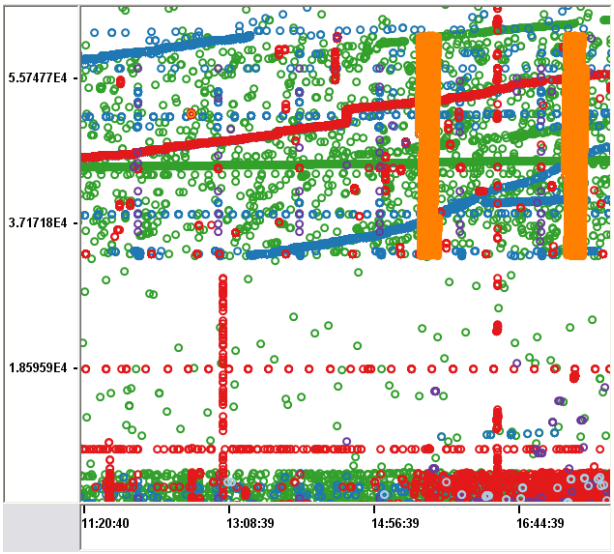


Fig. 2. Event diagram of source-port for the following traffic types: mail (green), DNS (blue), scan (red), SSH logins (purple), SSH attacks (orange), chat (indigo). Some interesting patterns are: (1) SSH attacks cycle over ports in the upper half of the port space (2) Servers that resolve host names for their logs via DNS create long sloping trails (3) scans exhibit a variety of different patterns

more evident: slow scans tend to have more extreme source to destination byte ratios than fast scans, and fast scans tend to send and receive less data than slow scans.

To evaluate the change in level of detail, we can calculate how many flow events are collapsed into each derived event for general traffic types. This technique very effectively reduces the amount of data that needs to be displayed.

Traffic Type	# flows / # derived event	# flows / # derived event
HTTP	6.47	375,563 / 58,019
Chat	52.9	12,537 / 237
Mail	1.22	10,178 / 8,355
Port scan	112.00	336 / 6
Port map	2.91	45,139 / 15,538
Scan	1167.13	60,691 / 52
SSH Attack	385.99	34,353 / 89
SSH Login	499.09	11,479 / 23
Web Crawl	580.98	72,622 / 125

6.3 Enhanced Filtering

In addition to supporting flexible filtering on event attributes, our system can also filter using pattern labels, and on patterns defined by logical clauses – analogous to modeling the pattern and then using the model to perform filtering.

To illustrate how filtering on pattern labels is more semantically meaningful, we will use residual analysis to identify traffic patterns. Let us assume that we are currently in the process of labeling network traffic, and have produced the event diagram of figure 5(a). Using the knowledge creation process, we produce a model of the pattern shown in figure 5(b) - which is mail, and apply it to generate labels over the data. We then compute the residual by filtering out traffic with label “mail” from figure 5(a) to yield figure 5(c), in which we can observe and model the pattern shown in figure 5(d) – which is a scan. Then in turn, we

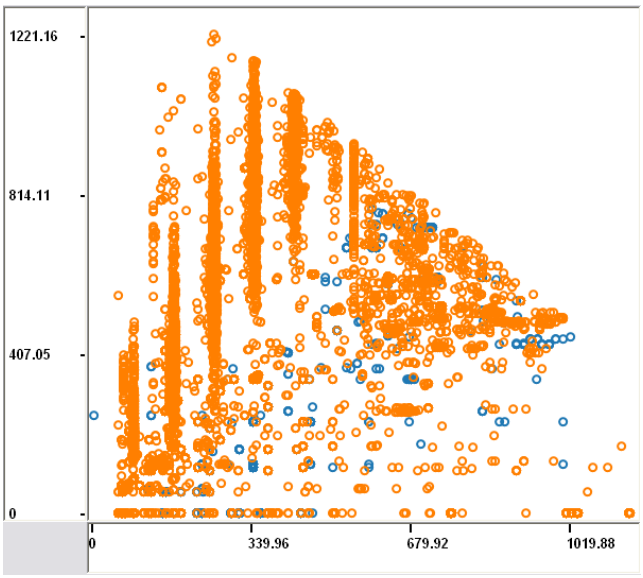


Fig. 3. Scatter plot of source bytes (Y) vs. destination bytes (X) for every flow in fast scans (blue) and slow scans (orange)

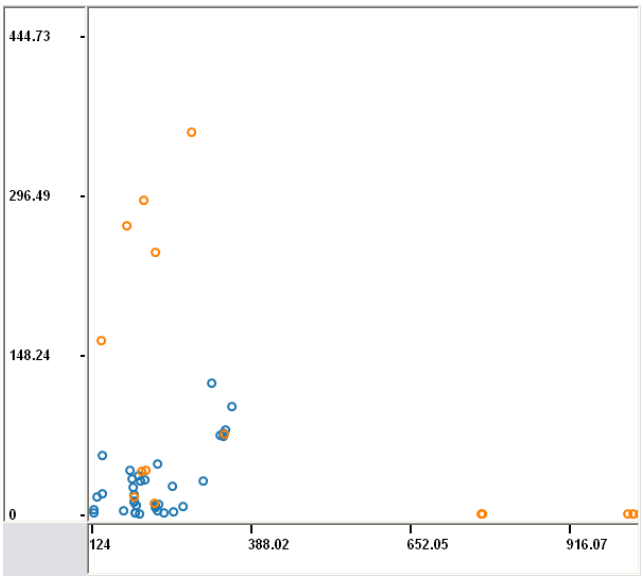
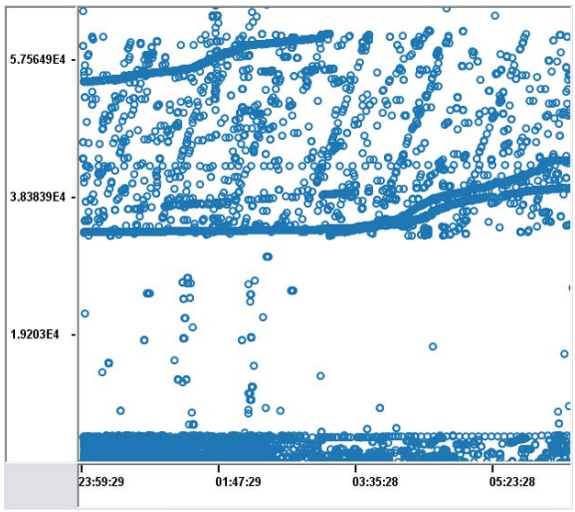


Fig. 4. Scatter plot of average source bytes (Y) vs. average destination bytes (X) for derived fast scan events (blue) and slow scan events (orange). One can clearly observe that slow scans tend to have either high destination bytes with low source bytes, or vice versa, while fast scans tend to have more balanced and lower source and destination bytes.

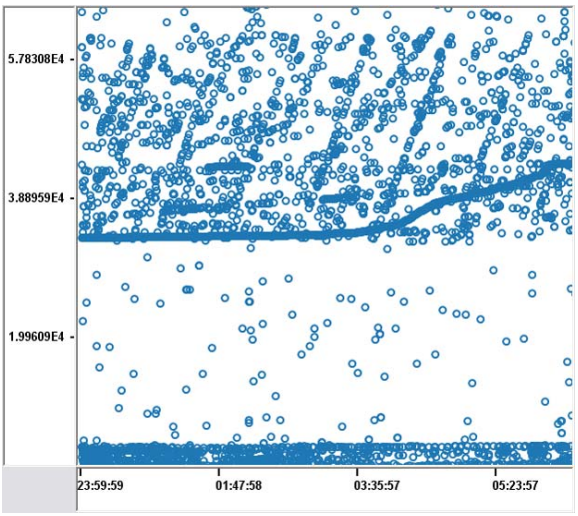
compute the residual by filtering out traffic with label “scan” from figure 5(c), to produce figure 5(d). This process can continue until we have labeled all the salient features in the visualization.

7 Evaluation

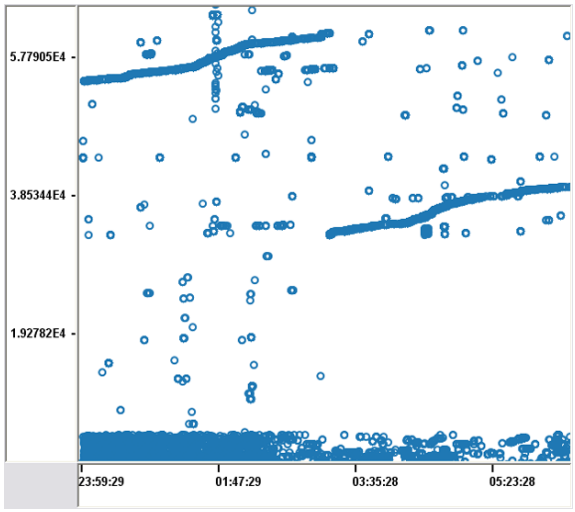
To evaluate the system, we attempted to label the network traffic for one day from our lab. This data was labeled by two authors, one with extensive experience administering the network.



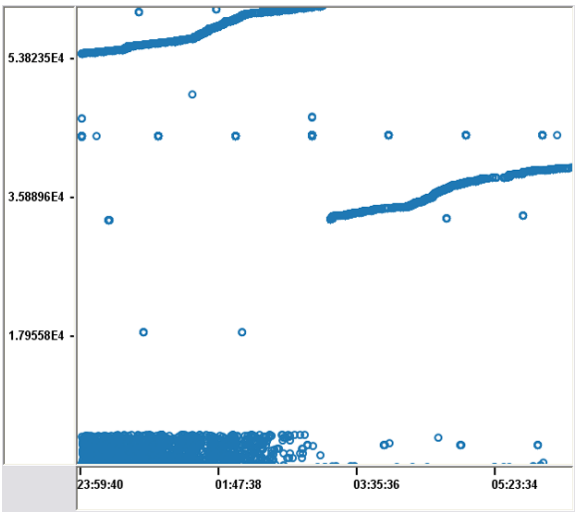
(a). Currently unidentified traffic.



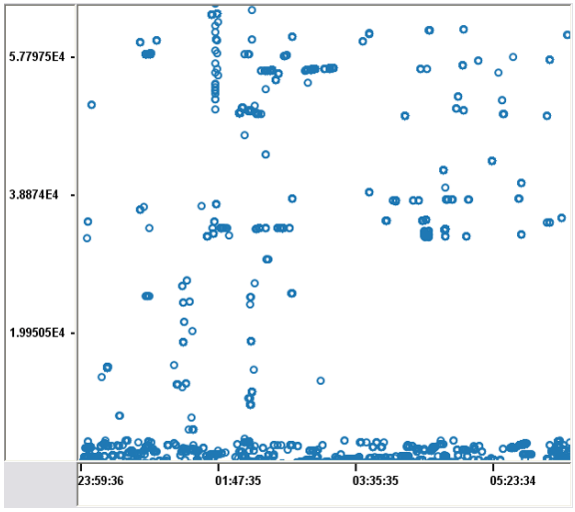
(b). Flows that correspond to mail.



(c). The residual flows after filtering out “mail” labels from (a).



(d). Flows that correspond to scans.



(e). The residual flows after filtering out “scan” labels from (c).

Fig. 5. Event diagrams showing flows with source port on (Y).

The primary goal of this experiment was to demonstrate the utility of using knowledge representation to aid the visual identification of traffic patterns. To test this hypothesis, we produce a set of models based on the identified patterns, and use them to classify the data. We were also interested in the accuracy of the resulting classification, and the overall usability of the system for this task.

The data that we used for this experiment was one day of network flows collected from an Argus sensor connected to the network switch in our laboratory. We collected approximately 1 million flow records for this experiment. Further statistics are shown below.

Property	Value
Number of flows	1,297,635
Number of local IP’s	1,065[1]
Number of remote IP’s	39,589
Total bytes sent and received	168,012,562,973
Total number of packets	207,121,399

[1] Includes spoofed addresses

Before we started the evaluation process, we discussed the types of network knowledge that we needed to represent with experts on network behavior. We then designed an initial set of predicates that is expressive enough to capture this knowledge, as well as efficient to compute. A subset of this initial set of predicates is shown in table 1. During the analysis process, we extended this set of predicates as we discovered new patterns that could not be described using the initial set. Several new built-in predicates were added during this experiment. This was meant to be a pilot study. A production system would have a much larger set of built-in predicates.

To discover traffic patterns, we used a variety of visual representations, although depictions of sequences of events in time were the most useful.

We used a structured analysis process. The first step was to look for instances of an interesting pattern. Once these instances have been identified, a model of the pattern is produced based on the observed instances, and generalized to the entire dataset. Then we sought new patterns and used them to produce new models. Not including the time spent designing the predicates and implementing efficient logical evaluation methods, the labeling process took around one week to complete.

The resulting set of traffic models consisted of the 21 label definitions listed below. Most of the models were simple conjunctions. Here are two examples:

```
Portmap = first_dest_port_111(x1,x2,…) AND
          identical_src_ip(x1,x2,…) AND
          identical_dest_ip(x1,x2,…) AND
          time_sequence_0.5s(x1,x2,…)

Fast Scan =  identical_src_ip(x1,x2,…) AND
          dest_ip_in_sequence(x1,x2,…) AND
          more_than_50_events(x1,x2,…) AND
          high_dest_ip_access_rate(x1,x2,…) AND
          (is_tcp(x1) AND is_tcp(x2) AND …) AND
          (low_duration(x1) AND low_duration(x2) AND
       …) AND
          (low_total_data(x1) AND low_total_data(x2) AND
       …)
```

The set of models produced was able to classify approximately 80% of the traffic in the dataset. In the classification process, we gave priority to voluminous and significant patterns. A complete list of models and the percentage of traffic of each type is shown in table 2.

To quantify the accuracy of the classification precisely, we would need an independent way of classifying the traffic on our network, which we unfortunately do not have. Instead, we applied the models over the entire dataset, and performed extensive visual inspection on the classified traffic, and hand verified patterns by examining the raw data they were derived from. We did discover that in some cases the classification was wrong. These cases can be divided into the following 3 categories, in which the first contained the vast majority of cases:

1. The underlying data for the flows was wrong. In particular we were surprised to discover that Argus periodically drops data, and confuses the orientation of the flows.
2. Some traffic was misclassified by errors in our models. We were able easily correct these mistakes using the interactive clause creation technique.
3. Some traffic was misclassified because the knowledge representation was not sufficiently expressive. We attempted to correct these cases by tuning the model clauses; however, we found that this procedure only led to a trade off between false positives and false negatives.

Thus, in most cases we were able to rectify errors by improving the models.

These results are evidence that the analyst was able to utilize the knowledge representation to accurately identify traffic patterns, and that the models produced were faithful to his domain knowledge.

Table 2. List of models and percentage of traffic associated with each type.

Model Name	# Flows	% of Traffic
Chat	12,527	1.0 %
DNS	131,764	10.2 %
Fast scan	19,322	1.5 %
IMAP mail	15,885	1.2 %
LDAP	121,365	9.4 %
Microsoft file access	57,461	4.4 %
Multi source IP	74,577	5.7 %
NFS	62,103	4.8 %
NTP	35,027	2.7 %
Pop mail	1,763	0.1 %
Port scan	336	0.02 %
Portmap	45,273	3.5 %
Wrong direction	25,824	2.0 %
Send mail	19,322	1.5 %
Single source IP	63,732	4.9 %
Slow scan	41,639	3.2 %
SSH dictionary attack	34,353	2.6 %
SSH successful login	11,479	0.9 %
Success login traffic	23,605	1.8 %
Web crawl	72,622	5.6 %
Web page load	386,652	29.9 %

The performance of the system on a dataset of this size was fast enough to be used interactively although there are opportunities to improve the algorithms. It typically took less than 30 seconds to produce and update a visualization. Applying a model to the entire dataset took approximately 10 minutes.

We found that understanding network behaviors became easier as we iterated with the system. We believe this is because the main challenge is understanding the voluminous low level network traffic. Once the patterns in the traffic emerge, higher level application events are much more intuitive to understand. We also found that the process of interactive exploration stimulated the analyst to recall many facts that he was not able to remember before. We believe many of the rules and patterns used to classify the traffic could not have been produced without an interactive system.

8 DISCUSSION

In this section we will focus on three points of discussion, first, the usability of the system, second, the effectiveness of the logical models, and finally, whether the approach can be generalized to other applications.

The performance of the system is dependent on the amount of data that is being examined. In the 1.2 million record dataset, most of the visual analysis tasks take less than 15 seconds, and applying models can take up to 10-15 minutes. The experience of the analyst would improve if the system were faster. There are various directions we are pursuing. Some logical predicates are expensive to compute; we intend to implement special-purpose algorithms to speed up these predicates. The performance would also improve if we had faster database technology.

Another option is to remove the more expensive predicates from the system, although this would limit the expressiveness of the knowledge representation. To efficiently evaluate models on a large dataset, we have already made compromises in the expressiveness of the language. For example, in the current implementation, all variable predicates must have an associated ordering, in which elements of a true grounding are contiguous; second, a variable predicate cannot be used with a fixed predicate (accepts a fixed number of arguments greater than 1) in the same clause. However, while these compromises yielded significant increases in efficiency, the language remained able to describe important network patterns, as we demonstrated.

A related problem is the binary nature of logic, which can not describe models with uncertainty. The misclassified traffic corresponding to false positive and false negatives are those very close to the decision boundary, and intuitively are those that the analyst is not confident about. To express this knowledge, we are investigating methods of using probabilistic reasoning. For example, Markov Logic Networks can be used to reason probabilistically from a first order logic representation [16].

While the current system uses the knowledge base to enhance network analysis using Gantt charts, event diagrams, and scatter plots; we believe that the flexible use of knowledge representation is a general technique that can improve other types of visualizations and applications. The effectiveness of the approach will ultimately depend on relationship between the types of patterns that can be shown, and the types of patterns that can be modeled. For example, within network traffic, modeling temporally related objects was particularly effective. While the current implementation uses first order logic for network traffic analysis, it is an example of the iterated analysis approach, which is independent of the knowledge representation, and can be tailored to different types of patterns and applications.

9 Conclusions and Future Work

We have presented a network traffic analysis system that supports the use of previous visual discoveries to enhance future visual analysis. In particular, we have shown how analysts using our system can build upon previous visual analysis discoveries to visually explore, and analyze more complex and subtle patterns.

Our plans for future work in extending this system revolve around 3 main directions. First, our current implementation supports only a small fraction of visualization techniques, and we would like to extend its visualization capabilities to make patterns more salient. Second, we would like to increase the expressiveness of the predicates, so that the analyst can describe more complex patterns. Finally, domain knowledge, and the patterns that are observed do not always follow the hard constraints of logic, and as such, we would like to extend the knowledge representation to allow for probabilistic reasoning.

Acknowledgements

This work was supported by the Stanford Regional Visualization and Analytics Center (RVAC). This Center is supported by the National Visualization and Analytics Center (NVAC(tm)), a U.S. Department of Homeland Security program operated by the Pacific Northwest National Laboratory (PNNL), a U.S. Department of Energy Office of Science laboratory.

Reference

[1] Arcsight. http://www.arcsight.com/whitepapers.htm, 2005

[2] Argus. http://www.qosient.com/argus/packets.htm, 2001

[3] R. Ball, G. Fink, and C. North. “Home-centric visualization of network traffic for security administration,” In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, 2004

[4] R.A. Becker, S.G. Eick, and A.R. Wilks. “Visualizing Network Data” In *IEEE Transactions on Visualization and Computer Graphics*, 1995

[5] S.K. Card, J.D. Mackinlay, and B. Shneiderman. *1999 Readings in Information Visualization: Using Vision to Think*, San Francisco : Morgan Kaufmann Publishers, 1999

[6] H.B. Enderton. *A Mathematical Introduction to Logic*, New York: Academic Press, 2001

[7] R. Erbacher. “Visual traffic monitoring and evaluation,” In Proceedings of the Conference on Internet Performance and Control of Network Systems II, 2001

[8] M.R. Genesereth, N.J. Nilsson. *Logic foundations of artificial intelligence*. San Francisco: Morgan Kaufmann Publishers, 1987

[9] J.R. Goodall, W. Lutters, P. Rheingans, and A. Komlodi. “Preserving the Big Picture: Visual Network Traffic Analysis with TNV”, in *IEEE Workshop on Visualization for Computer Security*, 2005

[10] D. Hilbert, W. Ackerman. Grundzüge der theoretischen Logik (Principles of Theoretical Logic), Springer-Verlag, 1928

[11] M. Lad, D. Massey, and L. Zhang. “Visualizing Internet Routing Dynamics using Link-Rank”, *UCLA Technical Report TR050010*, March 2005

[12] K. Lakkaraju, R. Bearavolu, A. Slagell, W. Yurcik, and S. North. “Closing-the-Loop in NvisionIP: Integrating Discovery and Search in Security Visualizations,” In *IEEE Workshop on Visualization for Computer Security*, 2005

[13] S. McCanne, V. Jacobson “The BSD Packet Filter: A New Architecture for User-level Packet Capture,” In *Winter USENIX conference*, 1993.

[14] T. Munzner, E. Hoffman, K. Claffy, and B. Fenner. “Visualizing the global topoloby of the Mbone,” In *IEEE Symposium on Information Visualization*, 1996

[15] Q1Labs. http://www.q1labs.com/resources/white_papers.html, 2005

[16] M. Richardson, and P. Domingos. “Markov Logic Networks,” *Technical Report*, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://www.cs.washington.edu/homes/pedrod/mln.pdf, 2004

[17] L. Xiao, J. Gerth, P. Hanrahan, "Creating Logic Clauses Dynamically From Pattern Selections," submitted to *IEEE Symposium on Information Visualization*, 2006

[18] W. Yurcik, K. Lakkaraju, James Barlow, and Jeff Rosendale. “A prototype tool for visual data mining of network traffic for intrusion detection”. In *3rd IEEE International Conference on Data Mining (ICDM) Workshop on Data Mining for Computer Security (DMSEC)*, 2003