Real-Time Visualization of Network Behaviors for Situational Awareness

Daniel M. Best   Shawn Bohn   Douglas Love   Adam Wynne   William A. Pike

Pacific Northwest National Laboratory
Richland, WA 99352, USA

{ Daniel.Best, Shawn.Bohn, Doug.Love, Adam.Wynne, William.Pike}@pnl.gov

ABSTRACT
Plentiful, complex, and dynamic data make understanding the state of an enterprise network difficult. Although visualization can help analysts understand baseline behaviors in network traffic and identify off-normal events, visual analysis systems often do not scale well to operational data volumes (in the hundreds of millions to billions of transactions per day) nor to analysis of emergent trends in real-time data. We present a system that combines multiple, complementary visualization techniques coupled with in-stream analytics, behavioral modeling of network actors, and a high-throughput processing platform called MeDICi. This system provides situational understanding of real-time network activity to help analysts take proactive response steps. We have developed these techniques using requirements gathered from the government users for which the tools are being developed. By linking multiple visualization tools to a streaming analytic pipeline, and designing each tool to support a particular kind of analysis (from high-level awareness to detailed investigation), analysts can understand the behavior of a network across multiple levels of abstraction.

Categories and Subject Descriptors
I.6 [**Simulation and Modeling**]: General.

H.4.2 [**Information Systems Applications**]

General Terms
Algorithms, Performance, Design, Security.

Keywords
Visualization, High-Throughput, Real-Time, Symbolic aggregate approximation, Network flow, Behavior modeling.

1. INTRODUCTION
This paper presents a pair of network traffic analysis tools coupled to a computational architecture that enables the high-throughput, real-time visual analysis of network activity. The streaming data pipeline to which the tools we introduce are connected is designed to be easily extensible, allowing new tools to subscribe to data and add their own in-stream analytics. The visual analysis tools themselves— Correlation Layers for Information Query and Exploration (CLIQUE) and Traffic Circle—provide complementary views of network activity designed to support the timely discovery of potential threats in network data. CLIQUE uses a behavioral modeling approach that learns the expected activity of actors (such as IP addresses or users) and collections of actors on a network, and compares current activity to this learned model to detect behavior-based anomalies. Traffic Circle is a raw network flow visualization tool that is architected to provide detailed views of very large volumes of traffic (in the hundreds of millions of records per view) in such a way that analysts can identify features that could otherwise be obfuscated through analysis of aggregates alone.

The tools we introduce here are designed to analyze network flows, a construct that aggregates individual packets into sessions that summarize the communication between two IP addresses over a particular combination of source and destination port. We use network flows because they are ubiquitous; most enterprises use a flow collection system as flows provide a useful, albeit imperfect, summary of network activity. In our case, the end user organizations for our analysis applications commonly use flows as an entry point into analysis to find interesting patterns, features, or trends.

VizSec '10, September 14, 2010, Ottawa, Ontario, Canada.

Copyright 2010 ACM 978-1-4503-0013-1/10/09…$10.00.

2. MOTIVATION
Ensuring the security of a computing network requires the timely discovery, and ideally prediction and prevention, of off-normal events that can represent threats to information or infrastructure. While rule-based techniques can assist in detecting instances of known malicious activity, the fundamental premise of visual analysis is that human analysts are by necessity part of the discovery, resolution, and response process. Visual interfaces are therefore necessary to provide analysts with the means to assess patterns, identify data features that may be hard to detect except through human perception, and apply judgment to determine whether an anomaly is malicious or benign. However, three primary challenges complicate visual analysis of computer network traffic: time sensitivity, data volume, and focusing analytic effort.

2.1 Time Sensitivity
Network security is inherently a time-sensitive endeavor. Analysts are faced with the challenge of identifying potential threats or vulnerabilities as early as possible, so that significant compromise of data or resources is prevented or mitigated. Frequently, data collection and analysis infrastructures are

designed to support only forensic analyses of data that may be hours or days old by the time it is reviewed by analysts. Visualization tools often reinforce this analysis approach by operating solely in a batch mode, allowing analysts to issue queries against a repository to retrieve historical information. Tools like intrusion detection systems (IDSs) assist in real-time analysis by providing alerts based on predefined conditions (useful for detecting events like SYN flooding or null HTTP headers). The signatures on which these alerts are based are frequently static and, although they can be useful indicators for malicious activity, do not effectively summarize the state of a network for an analyst. The post-mortem analysis that batch-mode visualization enables often occurs too late for the analyst to take proactive measures; damage may have already been done and the network is in triage mode.

The goal of most of the analysis groups with which we work is to analyze data as soon after its collection as possible. This does not mean that every network flow needs to be reviewed by an analyst. Rather, our analysts seek broad awareness of the state of their enterprise sufficient to give them confidence that current activities are within bounds of acceptability and help them identify where potential problems might be incipient. Creating such awareness requires that summaries of network activity be calculated in real time and that visual displays capable of communicating changes in real time be created. When analysts perceive patterns or trends of concern, they need to be able to drill into the summaries to recover additional details and explore the raw data from which these summaries were derived. Linking visualizations capable of summarizing aggregate events in vast amounts of network traffic succinctly with those that enable detailed exploration of raw data can give analysts the power not just to detect events quickly but also to resolve them efficiently.

2.2 Data Volume

The volume of data available for analysis is also a difficult problem to overcome. It is common for an enterprise network to generate thousands of flow records each second, producing hundreds of gigabytes or more per day. The data store from which an analysis may draw can therefore contain terabytes to petabytes of data. In some of our end user's organizations, there are tens of millions of unique IP addresses in just one month's traffic. (The problem of vast numbers of discrete actors becomes even more challenging in IPv6 addressing.) Data volume is a leading reason that analytic tools must work offline post mortem. The volume of data simply overtakes the analytic process. Not only must new data be examined (both automatically and manually) for potential threats, but historical data must be available for efficient retrieval when the analyst wants to review it for earlier indicators or compare it to current events.

Processing, visualizing, and interacting with data sets containing billions of records per day introduces specific analysis challenges. Visualization of discrete records is rarely an effective entry point into large data sets. Most systems for depicting raw event logs are only effective for data sets with tens of thousands to hundreds of thousands of records. Beyond these sizes, rendering times and occlusion cause challenges for visual analysis. Moreover, reducing large data sets down to subsets suitable for such tools means that very small portions of the source data are actually explored by an analyst; filters must be applied to either reduce the time period under study to one that is brief enough to contain just the amount of data a tool is capable of displaying, or to select flow records on the basis of a limited set of attribute-value pairs (such as particular port numbers). In either case, visualizing such subsets can remove the needed temporal and logical context of the flows under study. Performing discovery on small subsets is inefficient and, by preventing indicators only observable over broad data sets from being detected, quite difficult. The analytic process itself is hindered when analysts must continually "dip in" to their data repository to retrieve new subsets but are unable to keep all of the data of interest in view at once.

2.3 Focusing Analytic Effort

A third analytic challenge, and one that derives from the data volume problem, lies in giving analysts cues about potentially fruitful paths of investigation. Given the billions of transactions per day available for analysis, where should the analyst who is looking for new threats begin? Given a particular indicator detected by an IDS, where else might activities related to that potential threat lie? Anomaly detection at the level of a gateway or router can help point analysts toward activities of interest, but identifying statistically significant anomalies at the gateway can require large shifts in traffic that a savvy attacker will seek to avoid. Low and slow attacks that attempt to hide in normal-looking traffic become extremely difficult to detect.

Our analysts frequently ask for "jump-off" points for their work. Given a tip or a clue, analysts can drill into the data and judge the potential impact of a threat. Generating these tips in a manner that minimizes false positives and maximizes return on analytic investment is critical for the successful adoption of visualization tools. One comment we heard many times when talking with analysts was the desire to know what is normal for their network. If they can efficiently compare current activities to known normal (or at least acceptable) profiles for users, hosts, or groups on their network, analysis of the deviations from expected behavior can give a strong starting point for investigation. More than just useful for anomaly identification, behavioral models also have a predictive capability. Using models to help predict future states can help organizations move beyond the "catch and patch" security posture common today. A new ability to proactively respond to nascent threats based on the combination of model predictions and observed activities can result.

3. MEDICI

3.1 Architecture Overview

At the Pacific Northwest National Laboratory, the applications we build are increasingly dependent on processing and visualizing data from multiple heterogeneous sensors and simulations. We have found that applications over multiple scientific and analytic domains have similar needs to ingest large volumes of data, perform complex and often computationally expensive analysis to reduce the data to some human-readable form, and deliver the results to multiple visualization displays. To ease the development and deployment of complex, high-performance analytic and scientific applications over multiple domains, we have created a single middleware platform, called MeDICi (Middleware for Data Intensive Computing).

MeDICi consists of three architectural elements:

1. The MeDICi Integration Framework (MIF), which provides a Java-based API, runtime, and associated tools for visually building and deploying complex processing pipelines
2. MeDICi Workflow, which provides a workflow designer to allow a scientist or analyst to visually create

an application from a set of components predefined in MIF

3. A facility for the capture and management of application metadata called MeDICi Provenance.

These components are loosely coupled and may be used in a number of combinations for any given application.

A typical usage scenario involving all three architectural elements is as follows: Software developers create processing logic by programming a set of cooperating modules and encapsulating them as components. These components are stored in a library so that they can be retrieved later. Each component, where appropriate for the application, takes advantage of API methods to capture and store provenance information in the repository. An analyst then draws upon this library of components using the Workflow Designer tool that allows her to query the store and drag components onto a canvas, visually creating a new application.

A central design principle of MeDICi is to incorporate existing open source and commercial off-the-shelf (COTS) software products in order to provide high-performance and enterprise-ready infrastructure. We then build simplification layers and value-add features that are specific to the domains we support. Thus, MIF is built on top of a widely used Enterprise Service Bus (ESB), Mule [1], which provides a multi-threaded service-oriented platform in which services (called modules in MIF) can communicate over virtually any communication protocol or messaging scheme. These modules can be easily "wired together" using a simplified Java API to create robust and modular event-driven processing pipelines. MeDICi Workflow improves usage of the OASIS standard Business Process Execution Language (BPEL) [2] by providing a simplified graphical language that is then transformed to standard BPEL and deployed to a BPEL engine. MeDICi Provenance uses an open source RDF repository to store meta-data about workflows, allowing scientists and analysts to review the exact conditions in which a processing result was obtained.

For the network visualization application being discussed here, the MIF API was used to build the analytic pipeline, without the use of the other two architectural elements; thus the remainder of the discussion will focus on the usage and implementation of MIF for this network visualization application. A more complete description of MeDICi, including details of each architectural element, can be found in [3].

3.2 Application Characteristics

We maintain that multiple visualization techniques with different strengths must be combined to create a full picture of network situational awareness. This approach requires an underlying distributed software architecture that is robust, scalable, and deployable in a production environment. Specifically, MIF supports these features required by cyber visualization applications:

- A wide array of visualization tools are supported, which can be easily added and removed from the system.
- Analytic routines that filter and aggregate data before it is visualized can be readily created and added to a processing pipeline.
- Multiple programming languages are supported so that existing or legacy programs can be incorporated into pipelines with new processing modules
- Multiple communication protocols are supported so that modules and programs running external to MIF can send messages over the most convenient channels.
- The underlying runtime environment is able to process data at a high rate and have the ability to scale horizontally to accommodate additional sensor streams and processing routines.

Also, MIF's flexible service bus supports the addition of arbitrary heterogeneous sensor streams. Therefore, this application can be extended to include additional sensors and processing routines. In fact, a predecessor to MIF was used to build a Distributed Intrusion Detection System (DIDS), which was deployed as a test application in a production environment at a large networked conference [4]. This system had many of the same features of enterprise performance and modularity. However, several improvements have since been made to MIF that make it even easier to construct a pipeline. For example, we have created a simplified Java API to aid the construction of a distributed application. We also have created a graphical component builder that allows the programmer to generate most of the infrastructure code needed to construct a pipeline while focusing on module implementation.

3.3 Application Construction

MIF provides a flexible framework for creating analytic workflows optimized for high-volume data streams such as network flows. In a MIF pipeline, in-stream analysis components consume data from upstream producers and provide derived results to downstream consumers. In a cyber analytics workflow (Figure 1), a MIF ingester pushes data from flow sensors through a processing pipeline that first transforms various flow record formats into the particular internal schema used by each analysis tool (allowing multiple third-party tools to be plumbed into the same data pipeline). Downstream consumers create summary statistics and aggregate traffic into adjustable time bins, both of which are used in the CLIQUE models described in Section 4. The pipeline also distributes data to the real-time visualization tools described in Sections 4 and 5; these tools implement a MeDICi listener that receives and processes flow records. Finally,

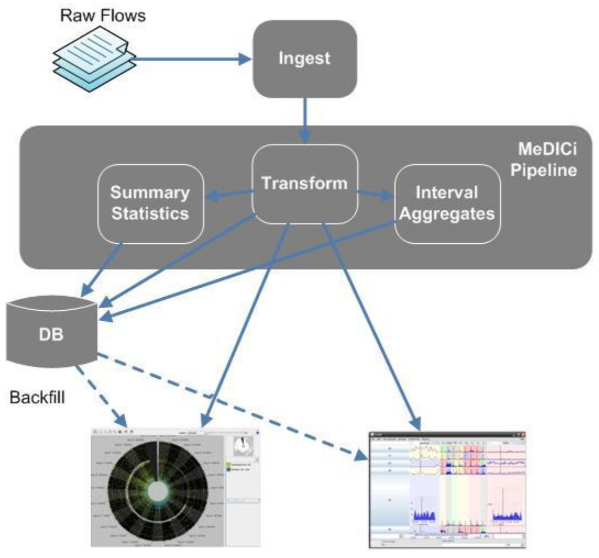


Figure 1 MeDICi pipeline

the pipeline stores the data that passes through it in a database that is used to backfill visualization tools with historical records when needed.

3.4 Application Performance

The performance of the MIF pipeline described here was measured against a dataset in which flow records were captured at a public event over seven days. For these tests, a day's worth of data from a particularly busy portion of the data was replayed against the described pipeline sped up to the maximum rate that the application could handle. This allowed us to test the bounds of the application and also provided a basis for a discussion on how the application could scale out to support much larger workloads. This application communicates with incoming data streams and visualization tools using Java Messaging Service (JMS), which is a widely used message broker specification that provides a publish-subscribe model of network communication. All incoming flow records and additional summarizations are stored in a relational database (RDMS). JMS, RDMS and ESB systems are all enterprise-strength technologies. However, each is challenging to tune for optimal performance on its own since each is highly sensitive to network topology, latency, and the intricacies of vendor-specific configuration options. This task is made even more difficult when an application is built on all three while communicating with an arbitrary number of visualization displays whose behavior may affect the overall system. Due to these difficulties, software vendors often generate high performance numbers by testing their products using trivial applications with short bursts of data. By contrast, the intention of these tests is not to produce the highest possible throughput numbers for an idealized MIF application, but to represent the performance of a reasonably well-tuned, real-world distributed network analysis system.

These tests were performed with three separate server-based software packages: MIF, the ActiveMQ JMS server, and a Postgres database running on a Dell 7500. This is an upper-end desktop machine, but also has a similar hardware configuration as compute cluster nodes in common use within our organization's current generation of state-of-the art commodity clusters. Therefore it is a good choice for a testing platform as it is also a likely deployment platform. Specifically, the computer has a dual quad core 2.80 GHz Intel Xeon® processor with 16 GB of memory. Although MIF does not require a large amount of memory, it does benefit from a large number of processor cores. This is because each module in a MIF pipeline (and the corresponding service running in Mule) is actually deployed as its own server with a dedicated thread pool. This allows each module to dynamically adapt to higher volumes of data by increasing the number of available threads. During the tests, CPU and memory usage was monitored to help determine where any bottlenecks occurred. In addition to the pipeline described above, two additional short programs were created: an ingester which simply reads flow records from files on disk and sends them at an increased rate over a JMS topic to the pipeline and a "performance monitor" which is a standalone application that receives messages from the pipeline and reports both an overall flow record processing rate (measured in records per second) and a rate for the most recent 30-second interval. The ingester and performance monitor were run on a separate machine from the MIF pipeline in order to ensure that their CPU usage did not interfere with that of the other components. Since each individual flow record is small (each record in this dataset averages 236 bytes), the ingester uses non-standard JMS configurations to improve performance: (1) transacted mode, in which groups of JMS messages are batched and sent as a single network message and (2) asynchronous mode in which the sender does not wait for acknowledgement of a message from the receiver before sending subsequent messages.

Since this is a real world dataset, the traffic volume varies greatly over the course of a day. Over the busiest four-hour period, the total average rate of flow records was 83 records/second, while during the busiest hour, the rate increased to 145 records/second. This system was first tested at this "actual maximum" rate, at which the system performs well and an observation of CPU usage shows that MIF, JMS, and RDMS are not taxed at all. Next, the maximum throughput was tested by doing runs in which the ingester is set to send data at increasing rates over several runs of the dataset. For the best run, the system reported a maximum average throughput of 2,781 records per second, which translates to processing over 240 million transactions per day on a single node. The highest throughput 30-second interval was 3,350 records per second. During this time, neither of the computers involved reached a CPU usage above 50%, leading us to conclude that the network may be the bottleneck in this case. Reasonable, although non-exhaustive, efforts were made to tune JMS performance and, to a lesser extent, Postgres performance as well. MIF performance was tuned by modifying the size of a module's thread pool and by changing the size of the JDBC connection pool that was used to insert records in the database.

Therefore, we maintain that it is likely possible to achieve greater performance by spending more time to tune each server product. However, it is also apparent that this system already performs quite well for its current task. To look at the throughput numbers another way, given the actual data rate from our flow sensor, our MIF pipeline could process a day's worth of data in 16 minutes — a 90x speedup over the actual rate. If this dataset represents a single typical site, these results imply that it would be possible to simultaneously support 89 additional sites at the actual data rate. Alternately, more complex automated data analysis could be moved from visualization displays to the service bus to improve client response time and make the analysis algorithms available to a wider array of client tools. Further, MIF is designed for scalability, so that the network analysis pipeline could be easily replicated to meet increasing data input volumes.

4. CLIQUE

Achieving real-time situational awareness of network activity requires techniques for summarizing large amounts of network traffic and for presenting those summaries in an easily understood interface. Although visualization of raw flows in large volumes can help analysts understand the current state of their networks and detect anomalous events, visual techniques that aggregate flows to higher-level abstractions can help analysts better cope with data scale. We have developed a behavioral summarization tool called CLIQUE, which bases its interface on LiveRac [5], that generates statistical models of expected network flow patterns for individual IP addresses or collections of IP addresses (each of which we term an "actor"), against which current activity is compared (Figure 2). CLIQUE uses MeDICi's ability to perform in-stream analyses to present real-time views of network behaviors.

The majority of changes made to LiveRac were done to the model and controller for the interface. Major changes include:

- Columns and rows were updated to feed off of data from the database (columns) and lists of user configurable groups (rows).
- Addition of the ability to listen to MeDICi's feed of data and update the interface without further database calls.
- Calculation and display of behavioral plots.

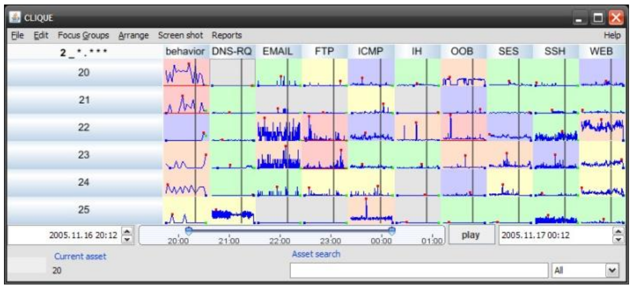


Figure 2 CLIQUE interface

The objective of CLIQUE is to help humans discover and detect potentially malicious events in vast amounts of streaming data. Previous research in this area has focused on two standard approaches for event identification in transactional data: signature-based methods and statistical anomaly detection [6]. Signature-based approaches are successful at identifying instances of known patterns, while anomaly-based approaches use general heuristics and statistical variances to identify patterns of interest. In practice, however, neither method alone is sufficient.

Recognizing that a gap exists between signature- and anomaly-based approaches, we implement a modeling approach that can be used in either a supervised or an unsupervised mode. Rather than producing visual displays that depict all of the raw data, we use a multi-level classifier and temporal model builder to reduce and condense the data visualized to an amount more suited to human interaction. The goal is to classify the data into patterns that represent categories of behavior inherent in computer network traffic. These patterns are then further classified, based on temporal sequence, to create higher-level abstractions of the activity in a network. This multi-level classification approach, coupled to a visual front end, allows the end user to view high-volume data in a much more condensed, information-rich fashion than is possible with raw flow visualization; this approach also attempts to capture the structure of an actor's traffic such that the behavioral representations serve as complete and useful summaries of the activities in which an actor is engaged.

Behavioral models such as those produced in CLIQUE are useful for anomaly detection, because they can be used to compare expected behavior against actual behavior. Rather than simply comparing expected traffic levels to observed signatures or applying a single policy-based model to each actor, CLIQUE behavioral models take a functional approach to anomaly identification. Such models learn the typical behavior of each actor over multiple kinds of activity (including port ranges and traffic types) and use these baseline representations as a predictor (over minutes, hours, days, or more) for that actor's traffic. These models are also helpful because they can also help identify what is *not* happening in an actor's traffic, which can be as useful an indicator as what *is* happening that shouldn't be. Times when the model predicts a certain kind of activity that doesn't occur can point to behavioral changes indicative of a potential threat (of course, adding contextual information to these models that indicates, for instance, that an actor is on undergoing maintenance can help resolve such anomalies).

4.1 Behavior Modeling

CLIQUE builds behavioral models for each actor on a network. Models are built in real time in response to user interactions, rather than being predefined. This approach allows models to be created for arbitrary collections of IP addresses on a network. The analyst configures CLIQUE initially by specifying groupings to use when CLIQUE launches. For instance, the analyst might configure CLIQUE to show behavioral summaries for

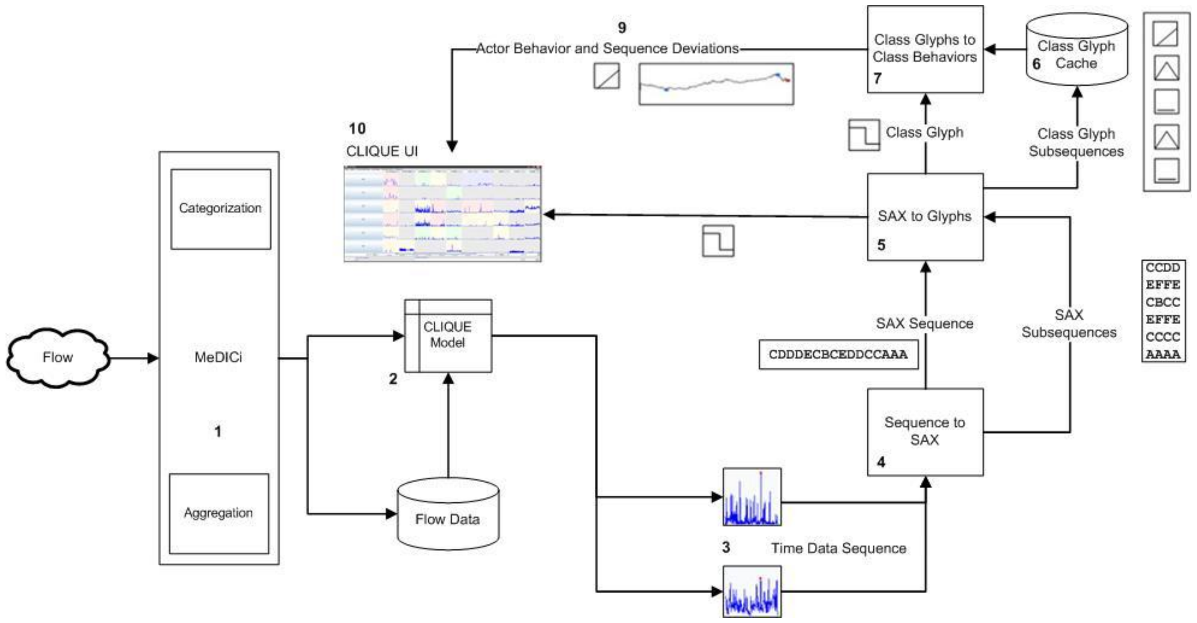


Figure 3 Behavior modeling process flow

aggregations of IP addresses such as buildings in an enterprise, subnets, or organizational units. Later, we describe how drill-down through the CLIQUE interface allows the analyst to decompose these actor groups in subgroups and (eventually) retrieve models for individual IP addresses.

Figure 3 outlines the process for creating behavioral models and calculating behavioral deviations. The behavior modeling process begins with identifying natural clusters of flow records. A streaming classifier within the MeDICi pipeline creates categories of flow records based on shared attributes (step 1 in Figure 3). We have implemented and reviewed the utility for a collection of such classifiers, including a modified version of QROCK [7], the VFDT algorithm [8], and a support vector machine (SVM). The QROCK classification approach uses training samples on the order of tens of millions of records to generate clusters of flows (typically hundreds of clusters). VFDT builds a decision tree from attributes in the flow records and assigns flows to categories made up of the leaf nodes in the tree. The SVM implementation is a supervised classifier that uses training samples from each of nine predefined categories of flow traffic (e.g., HTTP, email, FTP, SSH) that a team of analysts determined would be useful to track. We do not include source or destination IP addresses in any of our clustering algorithms, because we want to identify classes representative of all activity on a network and we later label each IP address with the classes it demonstrates.

One limitation of the SVM implementation is that it uses a relatively small number of predefined categories. With a larger number of more granular categories (such as could result from a semi-supervised or unsupervised classifier), we can describe the various kinds of traffic expressed through flows with more specificity.

Regardless of which classification algorithm is used in our pipeline, however, when each flow enters the pipeline it is assigned a class. For a specific IP or group of IPs, we sum the number of occurrences for each category at a user-defined temporal resolution. For the purposes of the current work, we selected an interval of 1 minute. This interval was chosen based on the amount of data available and the level of detail needed to be shown during development. To allow for other interval selections, the pipeline configuration could be updated or aggregation of other intervals could be done.

4.1.1 Symbolic Aggregate approXimation

Although we have classified each flow into one of the characteristic categories of activity on a network, we have not yet reduced the number of flows to be analyzed. To accomplish this reduction, we use a technique called SAX – Symbolic Aggregate approXimation that allows streaming dimensionality reduction and generation of a representation (word) based on time series discords [9-10]

The use of SAX begins in step 3 of Figure 3, where data is sent to the SAX module from the CLIQUE data model. For each channel (the temporal activity for a given category), the interval aggregations and the average historic interval aggregations are sent to the module. This provides the current and historic state for the channel at the visible time window. The output of the SAX module is a given actor's behavior. The actor's overall behavior can be considered the temporal activity for each category over time, akin to monitoring a multi-channel data recorder. We convert the stream of cluster labels applied to each actor's flows into a SAX representation by aggregating on the cluster label and using the values over time as input. The SAX representation generates a "word" that symbolizes the activity for a given channel (step 4 in Figure 3). This word can be generated by creating a plot of the activity level for a particular channel over time, then segmenting the plot into sections along the y-axis based on the normal distribution of activity (Figure 4). We then assign each y-axis segment a letter from alphabet defined by SAX. The

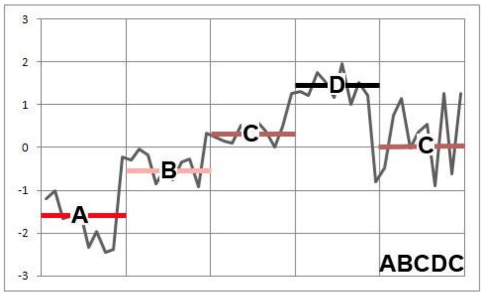


Figure 4 SAX time series conversion

alphabet is determined by the level of resolution desired. If only 4 characters are desired, then A, B, C, and D comprise the alphabet. For a given time interval, the series is segmented along the x-axis and each x-axis segment is assigned a character from the alphabet based on where the segment lies on the normal distribution curve (characters assigned to the y-axis segments). Combining the characters for each segment produces a word.

However, all of the words for all of the channels for each actor is still too much data, so we convert the SAX representation into one set of glyphs (step 5 in Figure 3). The intent of the glyphs is to approximate the symbolic representation for each channel even further. The transformation of a SAX representation to a set of glyphs defines a new vocabulary to represent the trends occurring within the data, such as remaining constant, decreasing, increasing, peaks, valleys, and stair stepping (Figure 5). This is accomplished by splitting the word generated by SAX into intervals; each interval is then converted to a glyph. The glyphs can be considered characters from the available glyph language.

This process is performed on each channel creating the set of glyphs for a given interval (or word). We then create a matrix that contains each channel for a given actor and each interval that a glyph has been created for. At the intersection of a channel and interval is the particular glyph for that channel during that interval. Each column now represents an actor's current behavior across all channels in glyph notation for that temporal sub-segment.

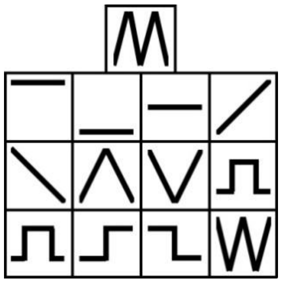


Figure 5 Glyphs used to encode SAX words

A channel of individual sub-segment glyphs characterizes a channel's activity and provides discrete insight into the behavior of the channel. Individual channel glyph data for all time sub-segments are passed to step 7 in Figure 3. Once we have the representations across all channels for a given actor, we are prepared to determine if deviations are occurring. To understand if deviations are occurring in the overall behavior, we employ steps 3 and 4 with the average of the data values for a similar time range.

By having two matrices, one of a historic timeframe and one of the current timeframe, we can compare these two glyph representations. The process to compare these representations requires us to treat the glyph representations as reduced syntax that can be compared via string comparison algorithms. We use the Edit Distance [11] algorithm to determine through a series of transformations how similar one sequence of glyphs is to another. By comparing the historic glyph representations with the current, a deviation from normal can be derived. By summing these deviations across all channels for a given actor per interval, we can determine over time in a holistic approach whether a series of interactions between the combined channels enable the system to detect deviations from the norm. This allows minor deviations in multiple channels to be shown as a larger overall deviation. The deviations are then sent back to the client (step 9 in Figure 3) where the deviation value for a given time interval is used to plot behavioral anomalies in the "behavior" column of the CLIQUE visualization. These deviations are also compared against alert thresholds that an analyst has previously established and control the color scheme used in the background of each cell.

4.2 Interaction

CLIQUE uses the LiveRac rubber-sheet interaction technique. To aid scalability; analysts can view highly generalized summaries of traffic in a heatmap style display. They then can zoom into a single heatmap cell to see greater detail in a variety of statistical charts; as analysts zoom in, they first see "sparklines" and then full data tables—the level of detail adapts to the level of zoom as seen in Figure 6. CLIQUE shows flow activity levels for each actor across a range of categories (such as web, ftp, and email) as well as a summary "behavioral" signal that reflects the deviation of that actor calculated from the SAX module (4.1). This behavior model, which can evolve over time, gives the analyst a sense of what is normal or expected for each actor.

Users can set thresholds in CLIQUE that alert them when an actor's traffic departs from expectations. For each category, a number of levels can be defined. Each level in turn can be assigned a value and color to encode that value. The threshold verification algorithm can be based on several factors in the data (min, max, mean, etc.). When the selected factor is within a threshold range, the background of the given cell is colored with the chosen encoding. This color-coding allows quick understanding of the current state for each category for each actor. For example in Figure 6 several cells are red, depicting issues while others green signifying a valid state. Behavior can also have a threshold encoding defined to alert users when deviations have gone past the threshold.

4.3 Results

A key attribute of the tool is the ability to determine the normal state of a given network. Any departure from normal triggers an alert that can be further investigated. If an actor is showing as normal yet channels are showing past thresholds it may be an indication that thresholds need to be reevaluated. One can observe the normal state of the network at a high level and drill into groups to view finer detail when needed.

5. TRAFFIC CIRCLE

Analysis of aggregates, as is performed in CLIQUE and other tools such as Isis [12] is one way to create visual representations that scale effectively. Depictions of summary statistics such as counts scale indefinitely as long as the underlying mechanism used to generate those statistics similarly scales. However, aggregates alone can occasionally obfuscate events in large data volumes. Sometimes, understanding the characteristics of

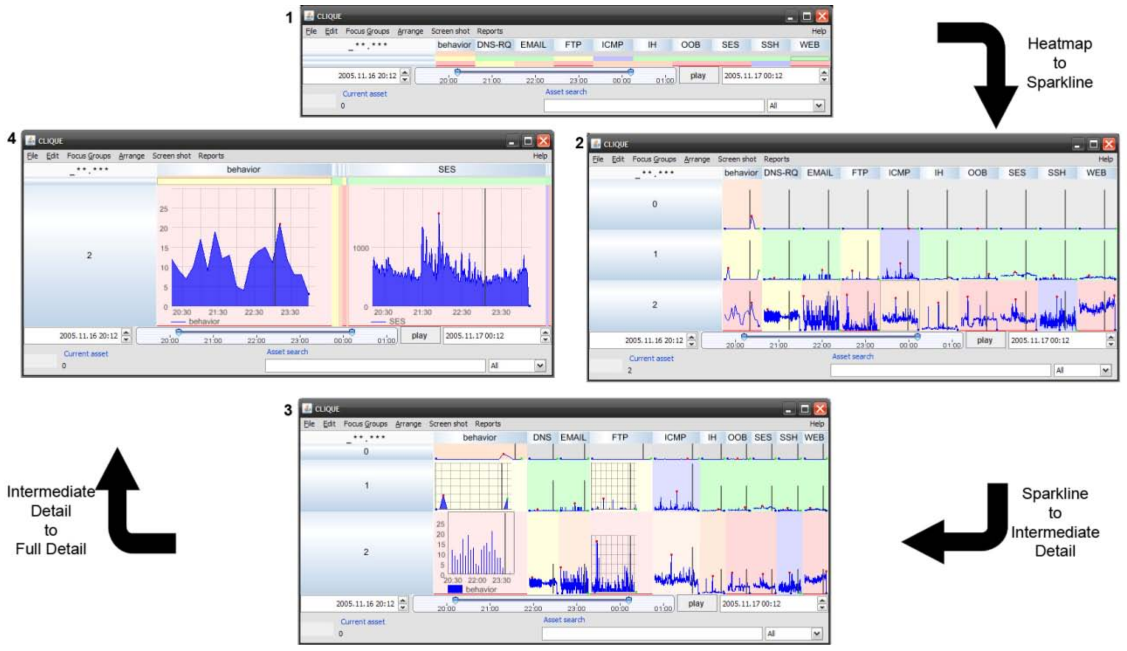

1
2
3
4
Heatmap to Sparkline
Sparkline to Intermediate Detail
Intermediate Detail to Full Detail
CLIQUE
File Edit Focus Groups Arrange Screen shot Reports Help
behavior DNS-RQ EMAIL FTP ICMP IH OOB SES SSH WEB
2005.11.16 20:12
2005.11.17 00:12
play
Current asset
Asset search
All

Figure 6 CLIQUE interface showing semantic zooming

individual transactions or their temporal pattern is necessary to resolve a behavioral anomaly discovered in CLIQUE.

To address this limitation, we have constructed a visual interface called Traffic Circle that complements CLIQUE by presenting detailed plots of individual flows. Our aim in developing Traffic Circle was to display as much flow data as possible in an interactive, exploratory interface. In their effort to understand “normal” traffic, our analysts want to learn as much as possible about the state of their networks, and raw plots of flow activity can help them learn what normal looks like and spot off-normal conditions. Typically, occlusion and rendering issues mean that plots of raw data may not scale beyond tens of thousands to hundreds of thousands of flows. This scaling challenge means that, in some enterprises, analysts can see at most a few seconds of traffic at once before the limits of the visualization tool are exceeded.

Our motivation in developing Traffic Circle is that, although summarization techniques like CLIQUE are effective at providing overviews of very large traffic volumes, there will always be features in data that the human perceptual system will be able to detect more readily than an automated system would—particularly those unexpected features that we would not have known to train an automated system to look for. By displaying large amounts of network traffic in an interactive plot—up to hundreds of millions of individual flows—we are able to support the deeper exploration of features initially discovered in CLIQUE and allow analysts to step back and see large segments of their network traffic transit their screen in real time. Such massively scalable plots introduce their own visualization challenges. Displaying large flow volumes on standard resolution displays can create occlusion problems, resulting in an undifferentiated mass of visual features that prevents the analyst from perceiving clear patterns. In addition, rendering and interacting with displays depicting millions or more flow volumes can be sluggish.

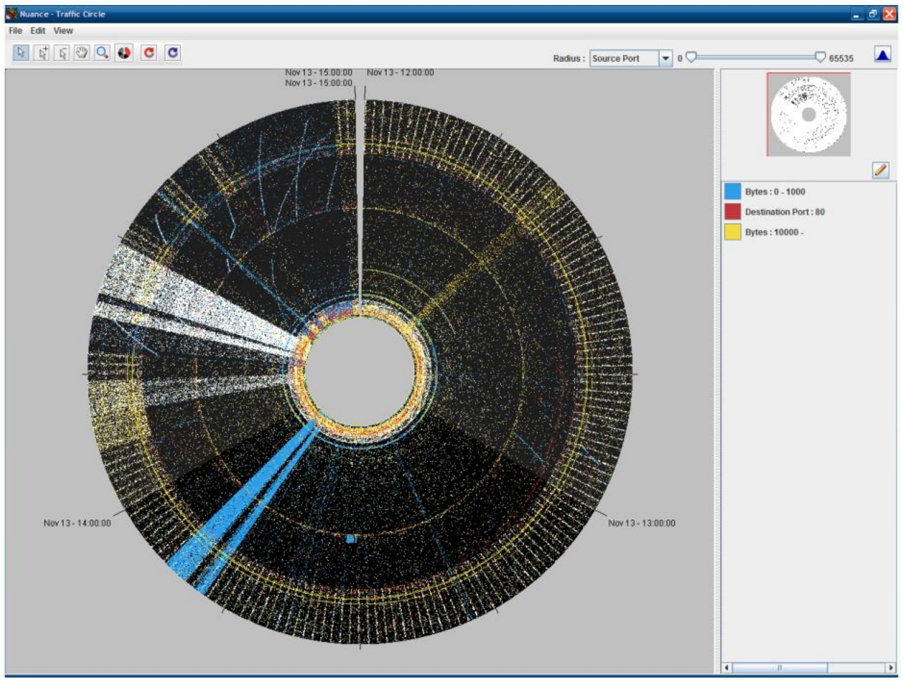
We have developed Traffic Circle to mitigate each of these challenges. To display large flow volumes on standard resolution, Traffic Circle includes a mechanism for assigning flows to “layers” based on common attributes; these layers can be toggled on and off to help reduce occlusion issues. Traffic Circle has also been successfully demonstrated on very large-resolution displays (up to 15.5M pixels), helping to reduce occlusion even when many layers are visible at once. To render and interact with millions of flow records, we leverage hyper-threading to render large data sets efficiently. Flows are partitioned across all available threads, with each thread responsible for drawing just those flows in its partition.

5.1 Interface

To the analysts with whom we worked in developing Traffic Circle, time is the primary attribute used to understand flow data. They wanted an easy way to perceive temporal patterns in their data, with the ability to drill down into very short time periods or scale up to very long time periods. Traffic Circle uses a circular “time wheel” metaphor to display flow records as arcs (Figure 7). Traffic Circle’s time wheel metaphor is consistent with an analog clock. The earliest time in the data set under examination is located at the top of the circle, to the right of the small gap. Time flows clockwise around the circle and ends at the top to the left of the gap.

Flows are ordered around the wheel by their start time, and their arc length corresponds to duration. The radial position of a flow is determined by a user-selectable attribute such as source or destination port or packet count. The attribute to which the radius maps is selected via a drop–down list at the top right of the user interface. Next to the drop-down list is a double-headed slider that enables the analyst to use a fisheye zoom capability to reveal additional detail where it is occluded (such as at the center of the circle). (A rectilinear plot option is also available and is discussed in Section 5.2.)

In addition to loading static data sets from the backfill database shown in Figure 1, Traffic Circle can also operate in real time by receiving flow records from the MeDICi pipeline. When in streaming mode, the circle slowly rotates clockwise, adding new



Radius : Source Port 0 65535

🟦 Bytes : 0 - 1000
🟥 Destination Port : 80
🟨 Bytes : 10000 -

Figure 7 Traffic Circle interface

data to the beginning and removing data from the end. In this mode, it serves as a live situational awareness display that gives the user a near-real-time display of current activities. We find it particularly effective as an ambient display in the analyst's workspace; by observing the plot's activity over a period of days, the analyst can learn what typical activity looks like and can therefore spot off-normal events readily. When occupying a secondary or tertiary display in the workspace, the analyst can use peripheral vision to assess whether the plot currently "looks like it should" for the flow sensors that are feeding it. When features appear that look off-normal, the analyst can monitor their evolution while performing other work.

5.2 Interaction

A key motivation for using a circular interface is the affordance it provides for direct manipulation to zoom in and out in time. Analysts can "spin the wheel" to adjust the time period shown (from seconds through years), without the need to readjust mouse position. Additionally, Traffic Circle draws upon the ability to spot periodic elements that spiral plots have been shown to provide (albeit a lesser degree due to being a single circle) [13]. This ability helps in identifying features such as beaconing or port scanning that could occur over very short or very long periods. Traffic Circle provides a natural method of time zooming; the user simply grabs a time label on the edge of the circle and drags it around the circle. Time can be added or removed from both ends of the visible data. The user grabs a time label on the right side of the circle to interact with the start time and drags clockwise to add time and counter-clockwise to remove time. The user grabs a label on the left side of the circle and drags counter-clockwise to add and clockwise to remove time from the end time.

Traffic Circle uses traditional pan/zoom functionality as well as a radius and time zoom. By default, the value of the radius at the center of the circle is zero and the value at the outside edge is the maximum data value for the selected parameter. For example, if the user has selected to order the radius by "source port" and the largest source port in the loaded data is 5000, then the outside edge of the radius corresponds to 5000 and the center to 0. The user can independently adjust the inner and outer radius values to be viewed within the current range causing the data to spread out across the radius and producing less overplotting. Data that are located near the center of the circle view are compressed as the circumference in the center of the circle is shorter than on the outside of the circle. To minimize this effect, users can turn on a fisheye mode that stretches the data to fill the circle as the radius sliders at the top right of the interface are adjusted. Traffic Circle can also reverse the order of the radius.

Analysts use filters based on flow parameters to color or hide certain kinds of traffic, reducing noise and revealing features that are otherwise occluded. The filter panel, on the right side of the interface, lists the filters that are currently active. The filters are ordered so that the top filter in the list takes priority over any filters below it. The data are rendered in order starting with the bottom filter, ensuring that the topmost filters are not occluded by data that matches a filter that is lower on the list. Each filter behaves like a layer, allowing the analyst to toggle the filter between three states: on, off, and hidden. In the "on" mode, the color corresponding to the filter is used to paint the flow arcs that match that filter. In the "off" mode, the arcs corresponding to the filter are drawn in their default white (uncolored). In the hidden mode, the arcs corresponding to the filter are hidden from view. By selectively hiding layers, analysts can wade through large amounts of traffic efficiently and can overcome occlusion problems resulting from overplotting.

A radial selection device allows the analyst to retrieve additional information on a flow collection depicted in Traffic Circle. Summary tables that aggregate flow characteristics for a given selection can be retrieved, and through those tables the analyst can drill down to raw flows.

While the main display mode of Traffic Circle is circular, there is also a rectilinear in which Traffic Circle displays data as a traditional scatterplot, with time on the x-axis and an analyst-selectable attribute of the flows on the y-axis. The usefulness and simplicity of rectilinear plots has made them a standard plot for network flow data. For some users the rectilinear plot option provides a more familiar interface, although it removes the "spinning" interaction affordance for time zooming found in the radial plot. The user must instead repeatedly pickup and reposition the mouse to adjust the time period. Both views of the data incorporate the layering and time selection needed for exploratory analysis.

The circular plot, by providing an alternative to a Cartesian plot, provides a pattern identification aid. Simply changing the way that data are portrayed for an analyst from a Cartesian to a radial plot can present features that encourage exploration. Certain features appear more readily in either the circular or the rectilinear mode.

5.3 Performance and Scalability

Traffic Circle has been operationally demonstrated at data volumes upward of 125 million flows per analysis session, and uses a multi-threaded architecture to provide interactive exploration of large data sets. We have used a high-performance database from Netezza to enable query responses in interactive time as the user manipulates the interface. The scalability of Traffic Circle means that analysis is not bound by small data extracts that may prevent the full picture of a threat from being understood.

Issues with plotting flow data in a circular view include render time and the length of the arcs at different radius positions. The amount of time to render arcs is greater than the amount of time to render lines. Comparably, this means that the circular view renders more slowly than the rectilinear view. To address this issue, Traffic Circle uses a multi-threaded drawing engine to minimize the render time and maintain interactivity at large data volumes.

Screen resolution and display size become relevant when visualizing one glyph per flow. As display size and resolution decrease, overplotting increases. Visualizing data sets containing more than 1 million records on a standard desktop display results in overplotting, sufficient to entirely obscure any discrete features. To address this challenge, we have begun experimenting with running Traffic Circle and CLIQUE on large, high-resolution displays. In the powerwall view shown in Figure 8 capable of rendering 15.5 million pixels, we are interacting with upwards of 5 million flows at once. The ability to detect visually individual features at this data volume becomes possible.

Figure 8 Traffic Circle rendered on a high-resolution Powerwall

6. CASE STUDY

The following scenario depicts a possible interaction with the tools described here to explore a cyber event. This scenario is informed by our interactions with analysts in cyber security organizations and captures how Traffic Circle and CLIQUE fit into analyst workflows. The ability of Traffic Circle and CLIQUE to support both real-time and forensic analyses enables analysts to detect current events and use historical data to understand them in context. In this scenario, both Traffic Circle and CLIQUE are installed on an analyst's workstation. The MeDICi framework has been configured to output data to the tools and a backend database.

The analysis begins with CLIQUE and Traffic Circle both operating in streaming mode. The analyst is going through her normal daily routine: investigating items, answering emails, and performing other tasks. On one of the displays, Traffic Circle and CLIQUE update with the current network traffic on a time interval defined by the analyst, in this case 1 minute. As the analyst works, she notices that the color encoding for the behavior cells in one of the IP address aggregations she monitors – for Building A on her organization's campus – has changed from green to yellow. [Analysts can use any aggregation of IP addresses they wish.] This change signals to the analyst that behavior has started to deviate from the normal calculated by the SAX algorithm. The analyst makes a note to watch the group, but does nothing yet. As time progresses, the encoding shifts to red, signaling that the behavior for that group is highly deviant from its norm.

She leaves this instance of CLIQUE open on her desktop to monitor real-time traffic and opens a second CLIQUE window to explore the new anomaly more closely. The analyst adjusts the CLIQUE timeline to incorporate a larger amount of time to show the progression of Building A's behavior. She drills down into the Building A group by double-clicking its row. CLIQUE expands the heatmap to show one row for each room in the building. The analyst can now see that the behavior cell for Room 10 is colored red, while the other rooms are mostly gray and blue – this indicates that Room 10 is causing the majority of the behavioral deviation for the group, starting at 7:00 AM. Looking at the individual traffic type charts for Room 10, the analyst notes that typical behavior comprises web traffic. However, today there seems to be a large amount of SESsion (SES) traffic on port 445. SES traffic represents valid session traffic that could not be placed within another category.

The analyst recalls a report that went out from Gartner about port 445 traffic and a vulnerability in Microsoft's Message Block Protocol (SMB) [14]. Due to what the tools are reporting, and the recent news, she decides to investigate further.

CLIQUE has provided the entry point to the problem; a given machine is acting outside of its normal behavior on port 445. To explore the data comprising this event in more detail, the analyst turns to Traffic Circle. Traffic Circle is running in real-time mode on her desktop, and she observes the patterns to determine whether this view of all traffic looks similar to its typical view. She applies a filter to highlight port 445 traffic for all machines on her network and observes that the density of flows on port 445 appears to be increasing over the past few hours. To investigate the early stages of this event, the analyst adjusts the time wheel to show the same time range as CLIQUE is displaying. She pauses, the streaming updates to Traffic Circle so that she can examine data from earlier today.

She filters the current Traffic Circle view, which had been showing all network flows from across her organization, to show just traffic whose source or destination is an IP address in the address space assigned to Building A. She loads a summary statistics table for the flows she has highlighted – those on port 445. Traffic Circle aggregates these statistics by remote IP address, so she sorts the table by the number of flows to each remote IP. She sees that the majority of port 445 traffic is going to a small handful of remote IPs.

The analyst decides to double check her assessment by examining the rest of the port range to determine if there is an increase in activity across the range or just with port 445. She toggles off the port 445 filter and sees that the remaining traffic seems as she expects it. The traffic appears to be distributed evenly across the port range with the majority of sessions being short in duration. She creates a new filter to contain this other traffic on its own layer, and toggles the new layer off to show just port 445 traffic. The analyst selects a particularly busy section of the circle and highlights the traffic.

Using the attribute space drop-down list, the analyst switches the radial axis to show local IP addresses as the radius of the circle. The previously highlighted flow records remain highlighted but are shown in the new attribute space. It is immediately apparent to the analyst that the majority of the traffic is along a single line in that attribute space, signifying that the port 445 traffic is all coming from the same source.

The analyst creates and applies a new filter for the remote IP range in question, filtering out all other flow data. The analyst can now explore what other traffic is going to that IP range. She also saves the collection of filters she has made, as a record of her work and as a discovery aid she can share with her team. She can pass this filter set on to a colleague who will explore the traffic between the local and remote IPs she identified in more detail.

7. RELATED WORK

As the number and sophistication of cyber attacks increase, the challenge for businesses to protect their networks has become more difficult. As a result, there have been several contributions to the domain of flow analysis and visualization to assist with the problem. Some, like Traffic Circle and CLIQUE, make use of flow data to create various charts depicting attributes of network traffic [15-17]. Others look at the problem a different way and show the connectivity of different systems, vulnerabilities, and the attack paths that can be derived from that data [18-20]. A variety of complementary approaches should be used in any operational activity.

Neither Traffic Circle nor CLIQUE display network connectivity graphs showing possible connections from one IP to another. The Bundle Diagram available in FloVis [20] is one such tool that visualizes flows between IP addresses by bundling these edges together on a circular plot. By bundling the edges, FloVis simplifies the visualization and minimizes overplotting. The input to FloVis is processed SiLK flows that have been placed into a relational database, rather than real-time data as in our approach.

Another connectivity graph, used to visualize attacks, is VisFlowConnect [21], which does not bundle edges such as FloVis**.** However, this system does have the ability to accept streaming NetFlow records from a socket. Because of the streaming nature of VisFlowConnect, it could easily be incorporated into the MeDICi architecture, adding another capability to the suite of tools all operating on the same data. Although the edge bundling in FloVis can be easier to navigate and find patterns in than a non-bundled plot, it may be harder to incorporate such plots into our MeDICi architecture due to the preprocessing required.

Like Traffic Circle, many existing flow visualizations are 2D rectilinear graphs that show some attribute along the y axis, and time along the x axis. An example of this is the Activity Plot in FloVis which shows IP addresses and their activity over time [20]. The plot uses color to encode attributes to allow the user to see issues at a glance. Another use of this type of visualization is seen in Abdullah et al., which uses such plots to detect intrusion [15].

CLIQUE uses 2D rectilinear plots exclusively. Similar to Abdullah et al. the plots are mostly flow counts over time, with the addition of a behavior column. Intrusion is a subset of activity that can cause deviations from normal for a given actor. The FloVis Activity Plot uses a similar visualization as the heat map created by CLIQUE, which uses thresholds to determine color encoding. Changing the encoding to depict an activity metric in CLIQUE could be possible with minor changes.

A third type of visualization can be termed "non-standard" plots. Tools in this category would consist of visualizations such as Flodar [17], the follow on work of Flodar by Blake [22], NetBytes in FloVis [20], and Traffic Circle. Flodar and Blake use a platter visualization to show network traffic and the activity of a given IP address on that network. Blake contributes the ability to accommodate dynamic networks. The platter uses time as a radius and plots intervals toward the center of the platter. IP addresses are color encoded based on their role in the network. By remaining active the column (for an IP) remains on the outermost edge of the plot. The platter also visualizes data in 3 dimensions, having the height of a given column represent the amount of activity for a given IP.

NetBytes uses 3D space to visualize historical perspective instead of animation [20]. The purpose of using a static plot is to minimize change blindness that can be introduced with the use of animation.

8. FUTURE WORK

The behavioral modeling approach implemented in CLIQUE lends itself to other domains, and we intend to explore applications of the modeling technique in analyzing the transactional activity of Supervisory Control and Data Acquisition (SCADA) systems and the power grid. To improve our models' sensitivity and their ability to summarize accurately the activities of an individual actor, we would also like to incorporate other types of log files into our models. Records such as syslogs and other host application events can provide additional nuance to the models. The application of sequence detection techniques from bioinformatics can also be used to identify commonly recurring subsequences in our vocabulary (which might be benign), subsequences that are known to be indicative of malicious activity, and very rare subsequences (which might be suspicious new threats).

A further challenge with behavioral modeling is ensuring that behaviors that analysts do not want included in the model are removed from the source traffic. The risk of not doing so is that malicious activities are unwittingly learned as normal traffic. We are exploring techniques that allow analysts to flag certain suspicious traffic as traffic that should not to be incorporated into the model and to use the CLIQUE classifier in a semi-supervised mode that allows analysts to remove entire clusters of traffic that do not have a ready benign explanation.

Although CLIQUE and Traffic Circle operate over the same live data streams and backfill database, we anticipate coupling them more tightly. It should be possible to instantiate Traffic Circle directly from within a CLIQUE cell, launching a detailed flow chart from a CLIQUE aggregate.

9. CONCLUSION

Combining Traffic Circle and CLIQUE in a near real-time environment, provided by MeDICi, enables network support staff to visualize traffic as it occurs. By providing this capability, potential issues can be investigated as soon as behavior deviates from normal. We have discussed how CLIQUE can provide a jump-off point of instigation into other tools that do not abstract the level of information such as Traffic Circle. Traffic Circle enables an investigator to quickly view potential threats contained within raw flow records and apply many different attribute spaces and color encoded filters. The three tools together present an environment for defense in depth network visualization.

10. ACKNOWLEDGMENTS

This research was supported by the U.S. Department of Homeland Security Science and Technology Directorate. The authors are grateful to the analysts who shared requirements and evaluated prototypes for this work. The Pacific Northwest National Laboratory is managed for the U.S. Department of Energy by Battelle Memorial Institute under Contract DE-AC-76RL01830.

11. REFERENCES

[1] Mule Enterprise Service Bus: What is Mule ESB? Retrieved July 23, 2010, from MuleSoft: http://www.mulesoft.org/what-mule-esb

[2] OASIS Web Services Business Process Execution Language: Specification version 2.0. Retrieved July 23, 2010, from OASIS: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[3] Gorton, I., Wynne, A., Almquist, J., and Chatterton, J. 2008. The MeDICi Integration Framework: A Platform for High Performance Data Streaming Applications. In Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (February 18 - 21, 2008). WICSA'08. IEEE Computer Society, Washington, DC, 95-104. DOI= http://dx.doi.org/10.1109/WICSA.2008.21

[4] Wynne, A., Gorton, I., Almquist, J., Chatterton, J., and Thurman, D. 2008. A Flexible, High Performance Service-Oriented Architecture for Detecting Cyber Attacks. In Proceedings of the Proceedings of the 41st Annual Hawaii international Conference on System Sciences (January 07 - 10, 2008). HICSS. IEEE Computer Society, Washington, DC, 263. DOI=http://dx.doi.org/10.1109/HICSS.2008.19

[5] McLachlan, P., Munzner, T., Koutsofios, E., and North, S. 2008. LiveRAC: interactive visual exploration of system management time-series data. In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (Florence, Italy, April 05 - 10, 2008). CHI '08. ACM, New York, NY, 1483-1492. DOI= http://doi.acm.org/10.1145/1357054.1357286

[6] Cahill, M. H., Lambert, D., Pinheiro, J. C., and Sun, D. X. 2002. Detecting fraud in the real world. In Handbook of Massive Data Sets, J. Abello, P. M. Pardalos, and M. G. Resende, Eds. Kluwer Academic Publishers, Norwell, MA, 911-929.

[7] Dutta, M., Mahanta, A. K., and Pujari, A. K. 2005. QROCK: A quick version of the ROCK algorithm for clustering of categorical data. Pattern Recogn. Lett. 26, 15 (Nov. 2005), 2364-2373. DOI= http://dx.doi.org/10.1016/j.patrec.2005.04.008

[8] Domingos, P. and Hulten, G. 2000. Mining high-speed data streams. In Proceedings of the Sixth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Boston, Massachusetts, United States, August 20 - 23, 2000). KDD '00. ACM, New York, NY, 71-80. DOI= http://doi.acm.org/10.1145/347090.347107

[9] Keogh, E., Lin, J. and Fu, A. D. HOT SAX: Efficiently finding the most unusual time series subsequence. In Proceedings of the Fifth IEEE International Conference on Data Mining (November 27-30, 2005). ICDM'05. IEEE Computer Society, Washington, DC, 226-233. DOI= http://doi.ieeecomputersociety.org/10.1109/ICDM.2005.79

[10] Lin, J., Keogh, E., Lonardi, S., and Chiu, B. 2003. A symbolic representation of time series, with implications for streaming algorithms. In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (San Diego, California, June 13 - 13, 2003). DMKD '03. ACM, New York, NY, 2-11. DOI= http://doi.acm.org/10.1145/882082.882086

[11] Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady. 10 (1966), 707-710.

[12] Phan, D., Gerth, J., Lee, M., Paepcke, A. and Winograd, T. Visual analysis of network flow data with timelines and event plots. In Proceedings of Visualization for Computer Security (Sacramento, CA, October 29, 2007). VIZSEC'07. Springer-Verlag Berlin, Berlin, 85-99. DOI= http://dx.doi.org/10.1007/978-3-540-78243-8

[13] Weber, M., Alexa, M., and Müller, W. 2001. Visualizing Time-Series on Spirals. In Proceedings of the IEEE Symposium on information Visualization 2001 (October 22 - 23, 2001). INFOVIS'01. IEEE Computer Society, Washington, DC, 7.

[14] Pescatore, J. 1997. More Port 445 Activity Could Mean Security Trouble. Technical Report. Gartner, Stamford.

[15] Abdullah, K., Lee, A., Conti, G., and Copeland, J. A. Visualizing network data for intrusion detection. In Proceedings of the 2005 IEEE Workshop on Information Assurance and Security (US Military Academy, West Point, 2005) IEEE, New York, NY, USA, 2-3.

[16] Plonka, D. 2000. FlowScan: A Network Traffic Flow Reporting and Visualization Tool. In Proceedings of the 14th USENIX Conference on System Administration (New Orleans, Louisiana, December 03 - 08, 2000). System Administration Conference. USENIX Association, Berkeley, CA, 305-318.

[17] Swing, E. 1998. Flodar: Flow Visualization of Network Traffic. IEEE Comput. Graph. Appl. 18, 5 (Sep. 1998), 6-8. DOI= http://dx.doi.org/10.1109/38.708554

[18] Nanda, S. and Deo, N. A highly scalable model for network attack identification and path prediction. In Proceedings of the IEEE SoutheastCon (Richmond, VA, March 22-27, 2007). IEEE, New York, NY, 663-668.

[19] Noel, S., Jacobs, M., Kalapa, P., and Jajodia, S. 2005. Multiple Coordinated Views for Network Attack Graphs. In Proceedings of the IEEE Workshops on Visualization For Computer Security (October 26 - 26, 2005). VIZSEC. IEEE Computer Society, Washington, DC, 12. DOI= http://dx.doi.org/10.1109/VIZSEC.2005.14

[20] Taylor, T., Paterson, D., Glanfield, J., Gates, C., Brooks, and S., McHugh, J. FloVis: Flow Visualization System. In Proceedings of Cybersecurity Applications and Technologies Conference for Homeland Security (Washington, DC, March 03-04, 2009). CATCH'09. IEEE Computer Society, Los Alamitos, CA, USA, 186-198.

[21] Taylor, T., Paterson, D., Glanfield, J., Gates, C., Brooks, and S., McHugh, J. FloVis: Flow Visualization System. In Proceedings of Cybersecurity Applications and Technologies Conference for Homeland Security (Washington, DC, March 03-04, 2009). CATCH'09. IEEE Computer Society, Los Alamitos, CA, USA, 186-198.

[22] Blake, E. H. 2004. An Extended Platter Metaphor for Effective Reconfigurable Network Visualization. In Proceedings of the information Visualisation, Eighth international Conference (July 14 - 16, 2004). IV. IEEE Computer Society, Washington, DC, 752--757. DOI= http://dx.doi.org/10.1109/IV.2004.2