# Outlier Detection in Temporal Network Data

Kyle Boyer and Erik J. Sturcke

**Abstract**—We propose using a modified chord diagram to visualize temporally varying network traffic so that a network security analyst can detect anomalies in the traffic. The visualization represents network traffic via edges of the diagram connecting intranet hosts on the outer ring. External hosts are placed within the ring, also connected by edges. Line width is used to show traffic volume and color is used to show direction. This provides both high level situational awareness of traffic and the ability to see details for specific traffic once an anomaly is suspected, using zoom and filter techniques

**Index Terms**—Information visualization, network security, intrusion detection, anomaly detection

## Introduction and Motivation

In our current day society, the primary method for transferring data from place to place is via the internet. This is a wonderful convenience, and opens up immense possibilities for connectivity. However, it also introduces the problem of malicious activity. There are those that would use the internet in a malignant way to gain unauthorized access to data or transmissions. To combat this problem, network traffic must be analyzed. There are many tools and methods that already exist to filter and block potentially malicious traffic from achieving their intended result. These tools, to be effective, block traffic in real time. As such, they must be automated. They follow an algorithm to filter traffic and prevent its transmission. While this is a very useful, even mandatory, thing to have, it still raises the issue of indecisive traffic. There exist internet requests and answers that could be arbitrarily marked as malicious, or not. These particular activity instances might not be flagged as negative by an automated system, and yet they are still of interest as they could be suspicious. We define these cases as anomalous. In this paper, we present a method for addressing these anomalous cases, using a visualization of network data. This methodology takes advantage of the human brain as a decision maker, as opposed to a programmed algorithm. The depiction that we create is intended to be scrutinized by a network analyst. It is a tool that's intended use is assisting the analyst in addressing and dealing with these atypical, "strange" cases.

## 1 Related Work

The practice of visualizing network data has existed as long as there has been network data readily available. Even still, relatively speaking, Security Visualization is a young term. [5]. This contributes to the situation we are currently in. There has been such an explosion of data in such a short amount of time, in the internet age, that we are left with more data than we can process effectively. This applies to data about the network itself as well. There are far too many incidents and anomalies to try to dissect them individually. We need a better method. [3].

The method that has been proposed is to visualize the data, graphically. This has been done in a multitude of ways, with different effects and filters applied for clarity. For example, one of the more interesting approaches applies heuristic functions to filter the data in real time. [4]. This method emphasizes changes in the data, but discards outliers. This way of encoding the data takes advantage of a technique entitled "edge bundling." [1]. A geographic approach has also been suggested, but this method can only see large patches of data, rather than individual points. [3].

• *Kyle Boyer and Erik J. Sturcke are with University of Maryland, Baltimore County, e-mail: {kyleboy1,sturcke1}@umbc.edu.*

A large challenge with network data visualizations is the massive overload of incoming data. [4]. The main way that current visualization ideologies handle this is by filtering data, and combining trends. Edges are bundled, groups of anomalies are recorded, and individual data entries are discarded. This can be very beneficial, as it allows a user to focus on the data that 'matters.' Unfortunately, this comes with the side effect that lots of data is effectively lost.

Our project focuses on detecting anomalies in network data, similar to the work of Shiravi, Shiravi, and Ghorbani. However, the project that we propose emphasizes outliers. We want to be able to zero-in on outliers in the data, and scrutinize them, rather than discard them. Outliers in network data must result from some event. IE: something caused them to be there. Irregular behaviour is often the most interesting and valuable information in a network dataset. [3]. Currently, there are few systems that handle outliers elegantly, and are able to discern some sort of useful information from their presence. We hope to fill that void with our research and development.

## 2 Research Question

We must first clarify from our motivation that our area of interest for this project is temporal network data. We are not considering network topology, but temporally varying graphs of hosts that are communicating with each other. Our primary area of investigation is how visualization can enhance anomaly detection and evaluation of this kind of temporal graph data. We want to know how effective a visualization tool can be at performing this task.

## 3 Hypothesis

Prior to developing our prototype, we proposed a hypothesis to our work, based on foresight and previous work using data visualization tools. Our preliminary hypothesis is as follows. We propose that by utilizing a chord diagram and the ability of an analyst to choose visual encoding of attributes around the perimeter of the circle, certain anomalous traffic will be able to be visually detected.

## 4 Design Method

### 4.1 Data Characteristics

The data was taken from Mini Challenge 3 of the 2013 VAST challenge. The scenario for the data a corporate intranet with around 1000 hosts distributed at three different offices. Three kinds of data were included: 2 weeks of Netflow data with almost 70 million entries and 1 weeks work of intrusion prevention system (IPS) data with over 16 million events. The data also contained various host level health monitoring statistic taken at 5 minute intervals which we ended up not taking advantage of. The following summarize the data types for the attributes we used from each of the data sources.

#### 4.1.1 Host Data

| | |
|---|---|
| IP address | categorical |
| Office site | categorical |
| Type | categorical |

#### 4.1.2 Netflow Data

| | |
|---|---|
| Time | quantitative (interval) |
| Source IP | categorical |
| Destination IP | categorical |
| Source Port | categorical |
| Destination Port | categorical |
| Protocol | categorical |
| Payload sent | quantitative (ratio) |
| Payload received | quantitative (ratio) |

#### 4.1.3 IPS Data

| | |
|---|---|
| Time | quantitative (interval) |
| Source IP | categorical |
| Destination IP | categorical |
| Source Port | categorical |
| Destination Port | categorical |
| Protocol | categorical |
| Operation | categorical |

### 4.2 Task Abstraction

To successfully produce a visualization that would allow an analyst to make meaningful decisions about the data, one of our first objectives was to define our tasks, in terms of visualization goals. Of course, our primary concern is with outliers, with reference to netflow data.
This lead us to our first task: identify. Our tool should allow an analyst to discern points of the data that are of interest. This shouldn't be standard, routine data, and it also shouldn't be requests that will certainly get blocked by an automated program.

To do this, the distribution of the data is of paramount importance. By sensing information correlations, we can detect things that are potentially amiss. We want this occurrence to be easily tracked down by a user of the tool. To accomplish this, the data must be discovered, and subsequently explored.

### 4.3 Prototype

Our design originally centered around the usage of a traditional chord diagram. We planned to encode source and destination hosts around the edges of a chord diagram, and use the actual ribbons of the diagram to represent individual requests. In our prototype for the project, we followed this design principle. The results of that initial idea and method are displayed in figure 1.

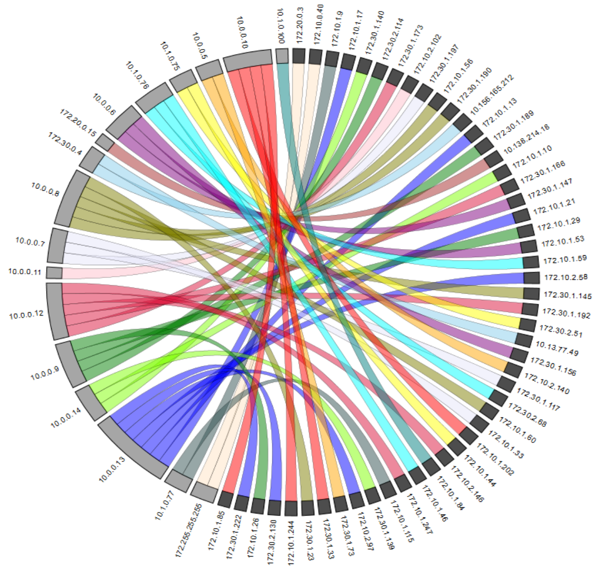

**Figure 1 Prototype chord diagram with traffic flows between hosts**

Source hosts are encoded on the right side of the diagram, in dark grey. The destination IPs are on the left side, in light grey. The internal ribbons each represent a single request. Each destination IP is encoded in a separate color for clarity and grouping. Our prototype design only displays 50 internal host connections. This is a very small fraction of our total dataset. When trying to load more data (1000 requests) into the visualization, it crashed. Additionally, even when only loading 50 instances into the program, the visualization looks very cluttered. Toward the center of the diagram, it is difficult to discern where any particular connection is going, due to the mass of overlapping ribbons. Due to these unforeseen issues, our design is not scalable at all, even to our test dataset. Clearly, it was back to the drawing board, as our initial plans for displaying the data wouldn't work.

After addressing these concerns, we came up with a modification to the chord diagram format. This one used straight lines instead of curved ribbons, and rather than connecting from one edge of the circle to the other, it separated internal hosts on the edges of the circle, and placed external hosts as nodes within the circle. This gave more space for the internal hosts because they could use the full circle. The inside of the circle could then be devoted not only to the edges representing the traffic, but also all external hosts that traffic is being sent to and from during the time window. We were able to effectively view far more data at once. Straight lines are far easier for the eye to trace than curved ones, and if all of the straight lines converge in one position, a "volume" can be inferred by the mass of lines.

This perceived grouping of lines also allows the user to see, at a glance, roughly how many requests were made from a particular source along with traffic volume. This is of particular importance when trying to discern anomalous behavior. One is able to see large spikes in traffic, potentially where they were not expecting to. Thus, our clutter problem is mostly resolved.

## 5 Implementation Method

The implementation continued with where the second prototype left off. Initially we set up a simple Mongo backed REST API which was then replaced with a PostgreSQL database with data served via a REST API using PostgREST offering better performance. We also did significant preprocessing on the data by collecting flows and IPS data into 1 minute buckets for each host pair and also high level 5 minute buckets to create the full overview. This allowed us to fetch 20 minutes of aggregated flows, or 50000 entries, in less than a second on a laptop. With these optimizations the back end was not the limiting factor.

The UI and visualization was written in Javascript using React as a framework. Although we did not use D3 for the data binding or DOM manipulation, we used it a utility library in scaling data. The SVG as well as other UI elements were drawn using React. Significantly more time was spent rendering the UI than fetching data from the backend. For 20 minutes work of data which took less than a second to load, the UI would take about 5 seconds to update on a laptop.

## 6 Results

Our finalized visualization, after rectifying the issues with the prototype and adjusting based on professional evaluation is displayed below.

One of the first things to notice about the product is the timeline display at the top of the image. The timeline displays the volume of data going into and out of the network for that particular day of the week. The white background corresponds to the 8am–5pm window and the gray off hours. This lets the user see how much traffic actually occurred during a particular day at a glance. More importantly, the end user can filter the data by clicking on a section within a day on the timeline, to automatically zoom the data display to that particular time interval. For instance, if the analyst clicks on Thursday in the timeline, in the center, the visualization will reload for the five minute window at noon on Thursday.

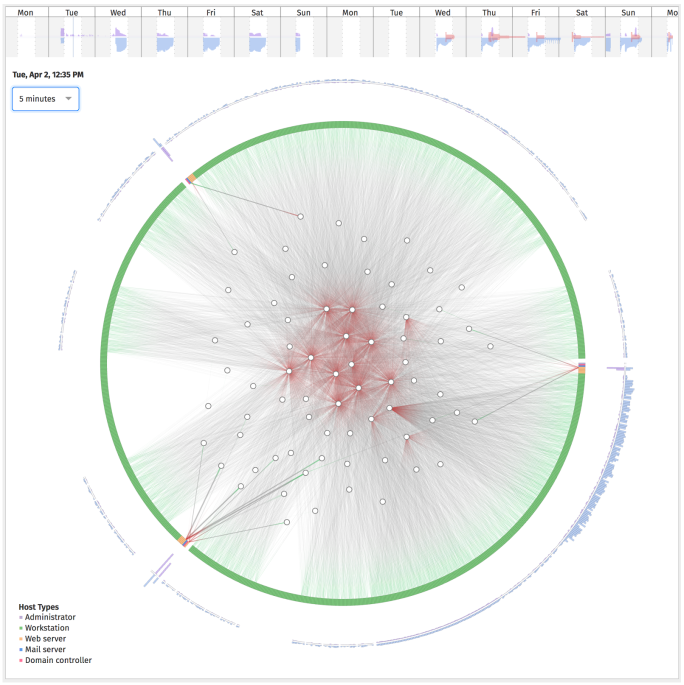

**Figure 2 Final visualization with traffic from internal hosts on the perimeter and external hosts inside the circle.**

Following that, we added a filtering mechanism based on time interval. By default, the visualization displays a five minute window of traffic. This interval can be changed to show more or less traffic at once, which is very helpful when trying to discern patterns.

Internal hosts are encoded around the edges of the circle, separated into local offices in three separate arcs. Every instance of traffic goes from one of the external host nodes in the center to the internal hosts in the outer edges. Along with the arcs that represent the hosts, there are blue volumetric arcs that encode the volume of traffic from a particular source. In the figure above, the lower right quadrant has a far higher concentrated volume of requests, as can be seen by the blue bars on the outer rim.

Finally, the thickness of the lines that connect the source and destination IPs encodes the actual traffic rates between those two hosts. A very thin line might represent only a very low traffic rate, while a very bold line represents a few hundred communications. The color shows the direction of the connection originating at green with a destination on the red side. The internal hosts are ordered by IP address. The external hosts are placed using a force directed graph technique to place external hosts closer to internal hosts they are communicating with to reduce edge overlaps.

## 7 Results Discussion and Analysis

For the scale of data that we were considering, we were able to create a visualization with reasonable performance even while displaying 50000 flows simultaneously across 1000 hosts. The REST API was certainly fast enough to keep up with the UI and rendering performance UI was sufficient although there are probably improvements that could be made.

The bigger questions is about scalability of the visualization. We a circle of radius 350px the circumference is around 2200px giving each host around 2px in width. This hints that the data we were using this project with 1000 hosts is right around the limit of this visualization. Also, without any kind of edge bundling or more aggressive aggregation, the flows inside of the circle become very dense when displaying on the order of 10000 flows. For our data this was in the 10 minute range. To scale this visualization further we would need to consider higher level overviews first avoiding showing individual hosts and flows with greater ability to zoom and filter.

## 8 Future Work

After the conclusion of our project period, which ends with the closing of the UMBC term in December 2016, we hope that our project will continue development. The UMBC semester put a very solid time constraint on us for the development of the project, and the release of this document. As such, the only things that aren't implemented are the intentions that we didn't have time for.

Overall, we're very happy with our product, but there are still a few improvements that we think could still enhance its usability, and its usefulness. Specifically, we would like to have more zooming and filter options, so that an analyst can more easily zero in on the data that is of interest to them at any particular time.

Additionally, even with the very thin lines, and randomly perturbed hosts, the visualization can become cluttered at times. To reduce that, and to give a more concise view of data that is "normal," we would like to implement the technique introduced in the related work section, called Edge Bundling. This would combine many of the host connections within the diagram into large "ribbons" to more easily show their correlation. Of course, this has the tradeoff of losing individual traffic, but we would only want to bundle traffic that is not important to the analyst anyway, just to reduce the clutter on the analyst's view.

Finally, The host placement within our diagram is chosen algorithmically, to attempt to give a good distribution of hosts across 2D space. This is done to reduce the clutter of the diagram, but does not have an actual encoding purpose. We would like to utilize the spatial position of the hosts within the centre of the diagram to encode a specific variable. This is still a topic of debate. The first logical solution would be to place the hosts based on geospatial position, tracked from IP address. However, as our test data set was a synthetic one, this wasn't practical for our display.

## 9 Potential impact

Our tool is not meant to be used as a replacement for automations. The intensity, speed, and volume of network traffic in the world simply does not facilitate being entirely filtered and dealt with by a human. The tool is intended to be used alongside these automated tools that already exist. We hope that our tool accomplishes two things.

Firstly, we want it to be used for its originally planned purpose. We would like the tool to be adapted for use in detecting behavior of interest that an automated tool might not catch. We want it to be a supplement. We hope that the tool will be used in this regard, and successfully help stop potentially malicious behavior that an automated filter did not catch.

Along the same vein, we want to demonstrate that visualization can be a useful tool for this purpose. As there has not been a large amount of research into the field of anomaly detection using visualization, we want to prove that it is worthy of further usage. If a party adopted our tool and put it to good use, it would prove that visualization is a viable option for detection of potentially malicious activity.

## References

[1] Holten, Danny. : Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data

[2] Livnat, Yarden. Argutter, Jim. Moon, Shaun. Foresti Stefano. : Visual Correlation for Situational Awareness

[3] Mansmann, Florian. Keim, Daniel. North, Stephen. Rexroad, Brian. Sheleheda, Daniel. : Visual Analysis of Network Traffic for Resource Planning, Interactive Monitoring, and Interpretation of Security Threats

[4] Shiravi, Hadi. Shiravi, Ali. Ghorbani, Ali. IDS Alert Visualization and Monitoring through Heuristic Host Selection

[5] Shiravi, Hadi. Shiravi, Ali. Ghorbani, Ali. A Survey of Visualization Systems for Network Security