A Tri-Linear Visualization for Network Anomaly Detection

Robert B. Whitaker

Utah State University
Department of Computer Science, UMC 4205
Logan, UT 84322

Robert F. Erbacher

Northwest Security Institute
Redmond, WA 98052
Robert.Erbacher@gmail.com

ABSTRACT

This research discusses a novel application of ternary plots to the visualization of network traffic data. These plots prove to be enormously effective at identifying anomalous network activity and can be valuable in monitoring network activity much more efficiently than can be done with existing techniques. The visualization was implemented in our existing visualization infrastructure to reduce development time. Testing was performed on actual network traffic data collected from a local network. Multiple anomalies were easily identifiable within the data set without any prior knowledge as to the contents of the test file. This paper discusses the ternary plot and its application to network traffic data, the formulas needed to calculate and display ternary coordinates, and the basic architecture for the visualization implementation.

Keywords: Security visualization, anomaly detection, ternary plot.

1 Introduction

Network managers and analysts currently have significant difficulties in keeping up with the number of malicious or malicious appearing events occurring on their networks. This is exacerbated by the scale of many of today's networks. With non-routable IP addresses, network managers can be dealing with tens of thousands of systems. These systems range from printers and routers to typical desktops, laptops, and the ever-increasing number of mobile devices such as iphones, portable game devices, kindles, etc. Currently, network analysts must acquire the set of malicious events occurring on the network, manually prioritize the events based on likely impact or some other internal criteria, analyze the events to identify their nature, and ultimately resolve the events.

This is a time consuming process that visualization techniques are designed to resolve. This process is further complicated due to the difficulty in determining which events are truly malicious and the fact that so many events are generally occurring at any one time on the network. Along these lines, current research has focused on the display of higher-level events or attributes, rather than the low-level network events. For instance, the representation of attack streams made up of the individual network events greatly reduces the problem scale and provides far more relevant and meaningful data. For this research, we are focusing on a novel visualization display designed intrinsically for the representation and identification of anomalous activity. This visualization can easily be extended to support large-scale networks, allowing for the removal or hiding of non-anomalous data; thus, removing clutter and obfuscation from the display.

1.1 Process Model Example

Consider, for instance, the need to monitor the network data for attacks at a primary router of an organization. Data on such a primary router can arrive at up to 1Gb/s or higher with current hardware and a single interface. Obviously the goal is to analyze the data and identify attacks, especially sophisticated attacks, within the arriving data. This process would entail analyzing available data, removing innocuous data from consideration, and identifying the details of identified attacks. Given the rate at which attacks are occurring [1] there is a need to analyze this volume of data continuously and rapidly.

With respect to the problem domain, visually representing even a subset of such large amounts of data simultaneously is not feasible. This would lead to enormous occlusion long before the entire data set could be represented. The visualization environment, therefore, must allow for subsets of the data to be rapidly displayed, analyzed, and resolved through direct manipulation. Resolution will necessarily incorporate filtering of resolved data elements; thus allowing the user to focus on the remaining or newly arriving elements. Resolution may simply be the determination that the data is either innocuous, a naïve attack of no concern, or a sophisticated attack requiring immediate measures.

1.2 Application Domain

Attacks on computer networks generally consist of five stages: reconnaissance, probing/scanning, attack, compromise/digging in, and migration. Given the sensitivity of many of today's networked computer systems, any form of successful compromise is unacceptable and considered too late in the protection and defense of such systems. For example, a compromise of 911 facilities, power plant control systems, etc. would be disastrous. Thus, detection and response must take place before a compromise can be completed. This requires that detection occur at the earlier stages, particularly during the scanning and attack stages. The reconnaissance stage generally takes place without the attacker accessing the target network. The attacker attempts to gain insight into the network organization and implementation in order to reduce the possibility of detection during the actual attack. Therefore, detection and prevention during the probing/scanning stage (or the earliest part of an attack) are the only viable means of thwarting an attack. This work focuses on the identification and analysis of network scans.

This differs significantly from prior work related to applying visualization to network security, which has focused on situational awareness [11][13], large-scale events [8][14], small-scale port analysis [4][8], IDS alarms [2], host processes [6], task-based approaches [16], etc. Instead, by building on previous work [5] the technique presented here has focused on detecting the low and slow scanning portion of an attack and the most sophisticated and challenging of such scans.

2 The Tri-Linear Visualization

The Tri-Linear Visualization is an extension of a ternary plot. A ternary plot is a type of graph that plots points within a triangular region, based on what percent of a whole matches each of three separate categories, as shown in Figure 1. While not nearly as common as other plots, such as a bar graph, or pie chart, ternary plots have been used for a number of purposes. A prime example of a ternary plot in relatively widespread use is that of the United States Department of Agriculture's soil texture triangle [17]. The soil texture triangle is used by the USDA and soil and plant scientists to label and categorize soil measurements. When using the soil texture triangle, soil is considered to have three attributes (silt, sand, and clay), each of which are a certain percentage of the whole.

The Tri-Linear Visualization plots points in a triangular region on the screen. Taking advantage of the ability to dynamically specify data mappings, a key component of our visualization infrastructure (termed AdviseAid), the points plotted in the Tri-Linear Visualization may represent virtually any aspect of a given data set.

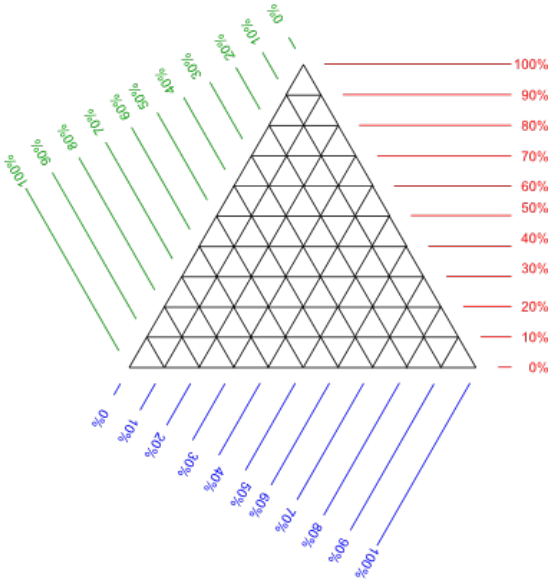

Figure 1: A basic ternary plot. Ternary plots measure three components that combine to form a whole. Based on the values along the three axes, a point can be plotted within the triangle representing the given percentages. The percentage points here are extended through the triangle to provide one example of zones. Values must be converted to percentages, for instance an IP address would be the IP address' 32-bit integer representation out of the maximum 32-bit value, i.e. 2^{32}-1.

By default, each point represents data transferred between two computers, so two IP addresses uniquely identify a connection. A node's position in the triangle can be mapped to a number of different attributes about the connection, and the node moves around over time, as the given attribute changes. The three attributes that we are specifically concerned with are port, protocol, and packet size, though our visualization infrastructure gives us the ability to map to other attributes of the data set as needed.

The Tri-Linear Visualization is primarily useful as an anomaly-based intrusion detection visualization, wherein the user is able to quickly and easily detect when something abnormal is happening. Anomaly-based [11] attack detection is known to be a very powerful form of detection because it allows a system to discover attacks that are not previously known. Current behavior is compared against "normal" behavior, and any time it deviates from the known normal behavior, the user is alerted that something may be wrong.

The Tri-Linear Visualization expands on the basic ternary plot in a number of ways. First, because the visualization occurs over time, the plot is animated. As time passes, nodes move around within the triangular plot area, and the user sees how the network is currently behaving, as well as how it might be changing.

Second, to make the changes more obvious, and to show the nodes' history, a trace line is drawn behind each node to illustrate the path it has taken.

Third, as a method of better illustrating normal behavior, as well as anomalous behavior, the triangular plot area is divided into a given number of smaller triangles called zones. All of the area within a single triangular zone represents similar data. Zones for a selected node are colored based on how much time the node has spent in the given zone. Colors are customizable, though the default is to color the zone black if the node has never spent any time in that zone, solid blue if the node has spent a great deal of time in the zone, and a shade in between if it has spent a smaller amount of time in the zone. If a node moves into a zone that it has never been in before, it will draw the user's attention to the anomaly.

A screenshot of the completed Tri-Linear Visualization is shown in Figure 2, illustrating the basic features of the visualization.

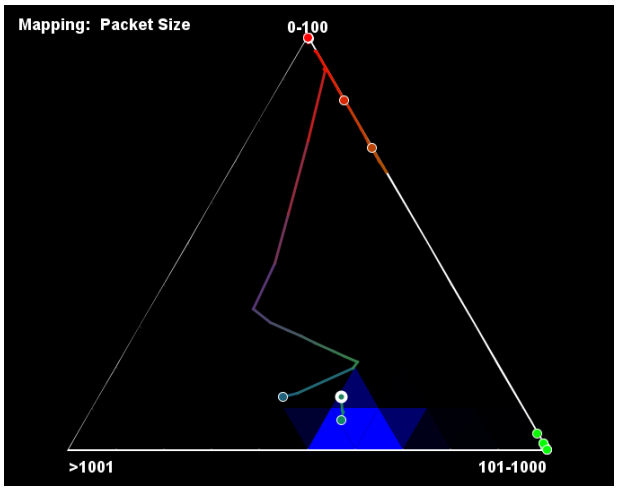

Figure 2: A screenshot of the Tri-Linear Visualization. This shows the basic plotting, along with traces and zones, colored in blue, for the selected node.

3 Previous Work

Detecting and analyzing attacks on a computer system is a difficult task. In fact, it may be impossible to come up with a perfect solution for this problem. There are a number of complicated problems that must be addressed in order to come up with an effective solution. First, on a network, or even just a single computer system, there is a massive amount of data that needs to be processed. Within any system designed for intrusion detection, an analyst must be able to process this data quickly, in real-time or near real-time, to be effective.

Second, on any given system, the majority of the data likely represents good or normal use of the computer network. An intrusion detection system or analysis tool must be able to effectively sort out data that represents an attack from data

that represents normal use of the system. Of particular interest here, are the concepts of false positives and false negatives. A false positive in this context is when an intrusion detection system identifies something as malicious activity when it is really normal or valid. A false negative occurs when a system indicates that something is normal, or fails to mark it as malicious, when, in fact, the event was part of an attack. Ideally, a system should have both low false positive and low false negative rates. It is generally thought that having a low false negative rate is somewhat more important, because false negatives mean that attacks are going by unseen, whereas false positives means the analyst has more data to look at, but nothing malicious is getting through. Thus, a good intrusion detection system or analysis tool has low error rates, especially low false positive rates.

Third, it is important for an intrusion detection system or analysis tool to be able to identify new attacks that have not been seen before. This is a critical feature of this type of system, because attackers are always coming up with new attacks and exploiting new vulnerabilities.

A variety of methods have been applied to the problem of intrusion detection. Kabiri and Gohrbani [10] point out some of the various types of techniques that have been applied in this area, including artificial intelligence, rule-based systems, data mining, Bayesian classification, and fuzzy logic. They point out that each of these methods has advantages and disadvantages that come into play in computer security.

One method for intrusion detection or analysis that has begun growing in usefulness is that of visualization. Ball et al. [3] point out that in the communications they have had with network analysts, the analysts have indicated that they have many tools to use, most of which are text-based. They state that the analysts are requesting more visual tools to help them quickly see the state of their networks.

Another trend in the field of intrusion detection is that of anomaly detection, rather than signature-based or rule-based systems [10]. This is because signature-based and rule-based systems essentially provide a blacklist of the different activities that constitute malicious activity. When new attacks appear, these systems generally miss the attack. Anomaly detection will most likely be able to detect the attack. Several anomaly-based intrusion detection systems have been, or are being developed. Depren et al. [4] for example, designed a system with anomaly detection and misuse detection, utilizing self-organizing maps (SOMs). A number of others have also applied anomaly detection to intrusion detection [11].

By combining these two trends, we can see that an anomaly-based visualization approach to intrusion detection and analysis may prove useful. The goal of this visualization is to allow for the identification of anomalies in a novel way. Additionally, the goal was to allow for the integration of multiple source parameters and allow temporal tracking of node behavior in a perceptually dominant display. When considering the design of a novel anomaly detection technique we must consider the implications of the fact that so much network activity *is* not normal.

The problem is that there will typically be so many anomalous activities that it is considered ineffective to perform anomaly detection. However, this can be resolved, as most anomalous activity should not get past the external firewall, assuming a default to deny policy. Additionally, while it may be time-consuming, all anomalous events need to be examined and either blocked or marked as innocuous. The transfer of hundreds of thousands of documents to wikileaks is the prime example of this. There were likely isolated anomalous events associated with the collection of this data before it was stored for distribution. While it would have been time consuming to examine all such events, the cost would have been far less than the impact of the release of the data. Finally, when considering what is anomalous we can consider that innocuous activity that looks anomalous will itself form a pattern. Once discerned, the goal will be to identify deviations from this pattern.

4 Methods and Implementation

The Tri-Linear Visualization's rendering uses Java's Swing and the Graphics2D class, as opposed to OpenGL. This was done to allow for rendering that is easy to work with and is aesthetically pleasing, without requiring substantial effort.

4.1 Overview

The basic structure of the Tri-Linear Visualization is straightforward, and essentially follows the Model-View-Controller design pattern. The architecture is illustrated in Figure 3.

The Tri-Linear Visualization takes advantage of many of the features that the visualization infrastructure provides, and as such, the visualization has no need to worry about input handling, other than to work with the data that the infrastructure gives it.

The underlying data model consists of a collection of nodes, which each represent a single point plotted in the Tri-Linear Visualization. Each individual node consists of three major pieces. First, there is some matching information, used to determine if new data belongs to the node or not. The code is designed so that the matching information is an interface, which allows developers to easily extend the visualization, but the default implementation allows for any network packet that has the same source and destination IP addresses to be considered a match. If no match is found in the set of nodes, a new node is created, with appropriate matching information for future use.

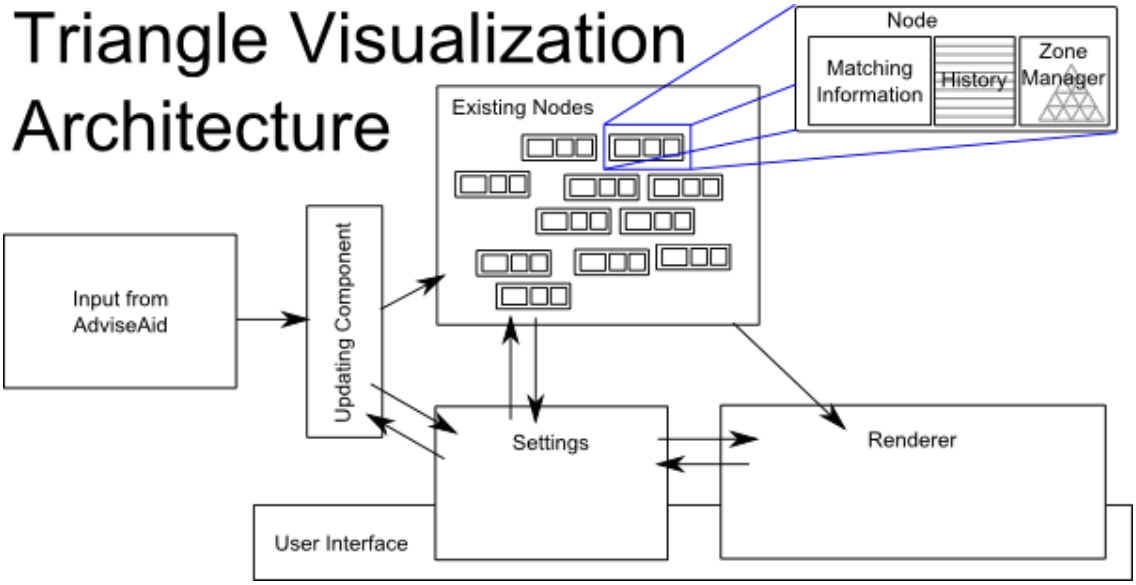

Figure 3: Illustration of the overall architecture of the Tri-Linear Visualization. Input comes in from AdviseAid (the visualization infrastructure), the underlying model is updated to include the new input, and the renderer displays the changes in the user interface.

When a new piece of data is added to a node, the node updates itself to include the new information. At the core of this process, the node must update its tri-linear coordinates, which determine its location. Additionally, a node contains a history of where it has been in the recent past. As enough time passes between updates, all of the nodes move to the next time step. The history of a node is stored as a queue, with new time steps being added to the back and the oldest time steps being removed from the front when the queue becomes filled to a given maximum capacity. Beyond the basic functionality of a queue, all time steps in the list are visible externally, so that the contents of the history can be displayed to the user on demand.

Each node also contains a zone manager, which takes the current location of the node within the plot and maps the value to a particular zone. A zone is a triangular region within the full triangular plot. Zones are a way of effectively grouping similar points together into a single bucket. As a node moves, it spends more time in particular zones than others. In most cases, a node spends all of its time within a single zone, or a small number of zones. When the node suddenly moves to a different zone, especially one that is drastically different from where it was before, it becomes obvious to the user visually that something abnormal has happened, and the node is no longer following normal behavior.

4.2 Tri-linear Coordinates

Tri-linear coordinates are a critical component of the Tri-Linear Visualization. They determine where a node should be plotted on the ternary plot. Tri-linear coordinates consist of three values x, y, and z, where $x + y + z = 1$. Tri-linear coordinates differ from 3D Cartesian coordinates in that the values combine to form a whole. While this provides the ability to plot ternary points on a graph, it is also a limiting factor in the sense that any of the components can be—and must be—determined by the other two values. For example, if $x = 0.2$ and $y = 0.3$, then we know z must be 0.5.

This has a direct impact on the way the visualization works, and is the largest limitation to a ternary plot. For example, if each node represents the flow of communication between any two IP addresses, the tri-linear coordinates, which tells us the location of a node, can represent the protocol used to communicate between the two. The x-coordinate represents the total percent of packets that use TCP. The y-coordinate represents the total percent of packets that use UDP. The z-coordinate represents the remainder of the packets, which are all of those that are using another protocol besides TCP or UDP, such as ICMP.

In some ternary plots, such as the USDA soil texture triangle, there are only three possible categories or labels, and each coordinate can represent a distinct category. For many other data sets, like the network traffic that we want to analyze, there are more than three categories. There are two methods of dealing with this. First, the third coordinate can simply represent all other data, giving us two distinct categories, and a third catchall category. Second, we can have three distinct categories, and anything that does not fit one of the three categories is removed from consideration. The Tri-Linear Visualization allows the analyst to specify which values are to be used for each category.

A second significant limitation of tri-linear coordinates is that each of the three categories must be independent of each other. Nothing that fits in the x category must fit into the *y* or *z* category. This restricts what kinds of categories can be chosen, to some degree, though most of the common categorizations should work without any trouble. For example, one could use the transport layer protocols TCP, UDP, and ICMP as the three categorizations. Nothing that uses TCP can also use UDP or ICMP.

4.3 Updating Nodes

One particular challenge with this visualization is how a node's tri-linear coordinates are updated, based on a new piece of data. A node contains a collection of data that determine the node's tri-linear coordinates. Because data within the visualization infrastructure is being processed as time passes, or at least in some form of sequential ordering, the tri-linear coordinate of the node is the average of each of the individual data items over time. For example, if two computers are communicating with each other and they send four TCP packets and then a UDP packet, the node's current location would be *(0.8, 0.2, 0.0)*. Thus with a basic implementation, a node's tri-linear coordinates is the average of the node's data over its entire history.

This type of basic implementation, however, has two big drawbacks. First, because it requires all of the node's history, as more time passes, more memory will be required of the visualization. Second, if a great deal of normal traffic occurs, and then suddenly a smaller amount of anomalous traffic appears, because there is so much more normal traffic, the node will barely change coordinates, and it will not produce a visible anomaly to the user.

An improvement on the basic naïve implementation is to only keep track of a user-specified number of recent packets, and average those instead, or alternatively, only keep track of all packets within a certain amount of time before removing them from the list. The former approach creates a maximum cap on the total amount of data stored, though the latter approach is likely to produce results that are more accurate. It does not guarantee a limited number of packets for each node, as the data set may contain a very large amount of data during a given timeframe. This data would be stored in its entirety, until enough time has elapsed, possibly causing the program to use up its available memory and slow the execution of the visualization.

While either of the two above approaches would be vast improvements over a naïve implementation that stores everything, neither of them were implemented. Instead, a hybrid approach was taken. The approach that was implemented in the current version of the Tri-Linear Visualization is a method wherein only a given number of packets are stored for the node, but the tri-linear coordinates of the node is calculated based on both the value of the new packets, and older packets, combined together with a weighting or decay factor.

$$t_{i+1} = \alpha\, t_i + (1 - \alpha) \sum_{k=1}^{n} v_k$$

Where:

t_i *is the tri-linear coordinate at the* i^{th} *timestep,*

α *is the decay factor, and* $0 \leq \alpha \leq 1$,

v_k *is the tri-linear coordinate of the* k^{th} *new piece of data, and* v_k *is either (1, 0, 0), (0, 1, 0), or (0, 0, 1),*

n *is the total number of new packets.*

We found that a decay factor of α = 0.995 works well, though this value is customizable, and other values may be better for different applications. In general, the closer α is to 1, the slower the node will move to reflect recent changes. With α = 1, the node would never move, and instead, retain its original position forever. The lower α is, the faster it will move. If α = 0, all old data is discarded immediately, and only the current data is used to determine the current location. This, in effect, degenerates to one of the previous methods described, wherein only a certain number of recent packets are used. Values near, but not at 1 tend to produce the best results.

This type of incremental change allows the visualization to be able to detect some forms of low and slow attacks. While old data may be completely thrown away, a summary of the data tracked numerically until a great deal of new data arrives. For instance, if a port scan is attempting to go undetected by only scanning one port every day, the Tri-Linear Visualization may still be able to detect it, because even though the data from days earlier has been removed, it is still having an impact on the node's location because of the decay factor.

4.4 Screen Coordinates and Tri-linear Coordinates

In order to be plotted on the screen, a tri-linear coordinate must first be converted from tri-linear coordinates to Cartesian coordinates in the screen's coordinate space. In Java's Swing, as well as virtually all other windowing systems, the top left corner of the screen or window has the coordinates (0, 0). The x-axis goes left to right, and the y-axis goes from top to bottom.

In order to plot the point, we must be able to take tri-linear coordinates and the given Cartesian coordinates of three points on an equilateral triangle to determine the screen coordinates of the given tri-linear point. Additionally, the Tri-Linear Visualization allows the user to select nodes in the graph. Since mouse events in Java are referenced by the screen coordinates of the mouse, we also need to be able to convert from Cartesian coordinates to tri-linear coordinates.

Mertie describes the process of converting tri-linear coordinates to Cartesian coordinates [15]. The Tri-Linear Visualization follows the basic formulas Mertie provides, with a few variations and modifications to match the screen coordinate system that we are using.

In this section, we refer to the main triangle that contains all points in the plot as *T*, and each of the vertices of the triangle as *T1*, *T2*, and *T3*, where T1 is the top vertex, T2 is the right vertex, and T3 is the left vertex of the equilateral triangle.

To convert a 2D screen coordinate to tri-linear coordinates, we follow a two-step process. First, the mouse coordinates (m_x, m_y) are converted to a unitized box surrounding an equilateral triangle with sides of length 1. Next, we determine a number of important measurements on the original triangle.

The following formulas are used to determine the values of *L*, *h*, and *a*:

$$L = T3_x - T2_x$$

$$a = \frac{\sqrt{3}}{6} L$$

$$h = 2a$$

To calculate the point (x, y), given (m_x, m_y), we use the following equation:

$$x = T3_x - \frac{L}{2} + m_x$$

$$y = m_y + T3_y - a$$

We can then calculate the tri-linear coordinates for the point (α, β, γ) with the following equations:

$$\alpha = \frac{1}{3}\left[\frac{2\,(p_3 - x\cos\omega_3 - y\sin\omega_3)}{h} + 1\right]$$

$$\beta = \frac{1}{3}\left[\frac{2\,(p_2 - x\cos\omega_2 - y\sin\omega_2)}{h} + 1\right]$$

$$\gamma = \frac{1}{3}\left[\frac{2\,(p_1 - x\cos\omega_1 - y\sin\omega_1)}{h} + 1\right]$$

$$where\ p1 = p2 = p3 = \frac{1}{3}$$

$$and\ \omega_1 = \frac{\pi}{3}, \omega_2 = \frac{5\pi}{6}, \omega_3 = \frac{3\pi}{2}$$

To convert from tri-linear coordinates to Cartesian screen coordinates, we use the formulas below:

$$x = +(T2_x - T3_x)\left(\frac{\beta}{1-\alpha}\right)$$

$$y = T1_y + (T2_y - T1_y)(1-\alpha)$$

Note that γ is not used in these calculations, because γ is dependent on α and β.

4.5 Zones and Tri-linear Coordinates

Zones in the Tri-Linear Visualization are small triangular regions within the main triangular plot area. All points within a zone have very similar tri-linear coordinates, and as such, zones have an important role in helping the user determine when the behavior of a node becomes anomalous. A node crossing into a zone in which it has not spent much or any time is an indication that something odd may be going on. It is especially so if the node makes a rapid change to a distant zone. The number of zones is user configurable but defaults to ten zones on each of the edges. A future goal will be to create zones automatically based on learned behavior. In essence, the movement of the nodes can be calculated; not including nodes exceeding the standard deviation of a majority of the nodes. Zones can then be created dynamically to contain each node or a group of nodes. A screenshot from the Tri-Linear Visualization is shown in Figure 4.

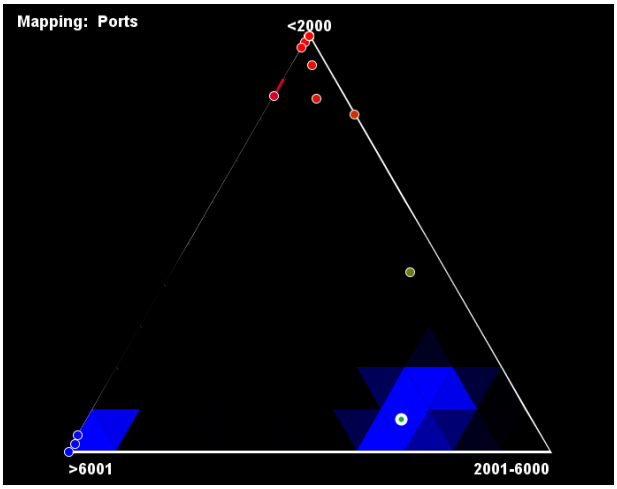

Figure 4: Screenshot of zones in the Tri-Linear Visualization. In this image, the zone data is for the selected point in the lower right part of the triangle. The zones indicate where the node has spent most of its time, with the color of the zone changing from black, representing zones that the node has never been in, to blue, where it has spent a long period of time. If a node moves into a black area, it is clear that something anomalous is happening, and is worth further investigation.

The current implementation of the Tri-Linear Visualization stores the zones in an array of arrays. Each row in the array represents a horizontal strip of zones across the triangle. Each row contains alternating orientations of triangles, upward oriented, then downward oriented. The first row is of size 1, containing only the single upward-oriented zone at the top of the triangle, the second is of size 3, containing the three zones in the second row of zones (two upward- and one downward-oriented triangles), the third contains 5 total zones, and so on, until the bottom of the triangle is reached.

As a part of this process, it is common to need to determine the zone of a node, given a tri-linear coordinate. This problem can be summarized as this: given a tri-linear coordinate (α, β, γ), and the total number of rows of zones in existence, determine the row and column in the triangular array for the zone that it belongs to.

To accomplish this, we first calculate the zone's value along each of the three axes, using the equations below:

$$x = (h-1) - (1-\alpha)h$$

$$y = (h-1) - (1-\beta)h$$

$$z = (h-1) - (1-\gamma)h$$

$$where\ h\ is\ the\ total\ number\ of\ rows\ in\ the\ zone\ array$$

Then, to lookup the actual row and column from these values, we use these equations:

$$row = (h-1) - x$$

$$column = \begin{cases} 2y & if\ y+z = row \\ 2y+1 & if\ y+z \neq row \end{cases}$$

Note that the function for determining the column becomes a piecewise function to account for both upward oriented and downward oriented zones.

5 Results

One interesting result that we noticed with the Tri-Linear Visualization is the ability to detect port scans. Port scans were easy to discover in an actual network capture file acquired from the local network. The Tri-Linear Visualization shows port scans as nodes that continue to move from one vertex to the other and back. If the visualization is configured effectively, the node will go around each of the vertices in a cycle. Figure 5 shows a screenshot of the Tri-Linear Visualization with two port scans visible. On the right is a port scan that is easy to see because the node continues to move back and forth from the top corner to the lower right corner. Upon further drill down, one sees that the scan starts at around port 1000 and incrementing by one on each step, up until around 4000, when it starts over. In the lower left corner is a second port scan that started somewhere in the 2001-6000 range, and continued to climb, and eventually moved the node over to the left vertex.

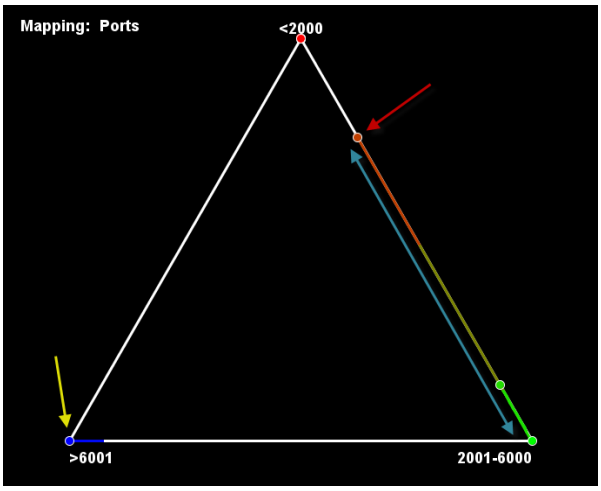

Figure 5: Two port scans in the Tri-Linear Visualization. Each vertex of the triangle represents different ranges of ports, with the low-end ports at the top vertex, the middle range ports on the right vertex, and the high range ports on the left vertex.

Near the end, several nodes suddenly turn extremely unstable, and move all over the plot. In particular, one node, shown in Figure 6, arcs through the middle of the plot indicating that it was communicating on a large number of ports. This shows a serious problem with the node. Because this node is selected, we can also see that this node had previously spent almost all of its time near the top vertex by looking at the marked zones in blue.

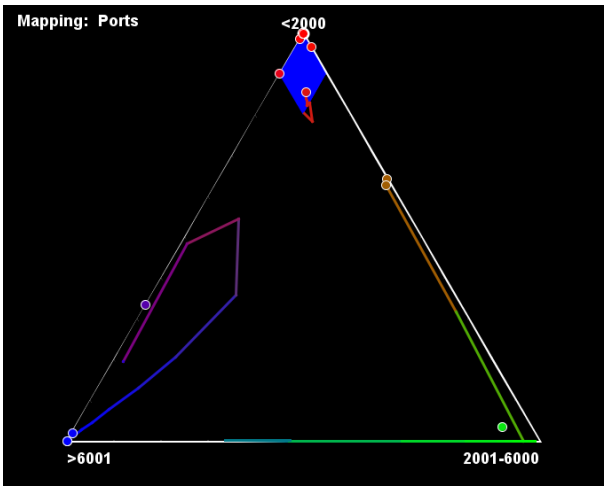

Figure 6: Anomalous activity shown in the Tri-Linear Visualization.

In a second example, shown in Figure 7, the visualization has the vertices mapped to packet size. In this example, the top vertex represents communication with small packets (less than 100 bytes long) the right vertex with medium sized packets (100 to 1000 bytes) and the left vertex represents communication with large packets (longer than 1000 bytes long) as would be normal during a large file transfer. During the analysis, we see two nodes that are constantly moving around near the lower middle part of the triangle. They move around together, which indicates that the two nodes are likely communicating with each other.

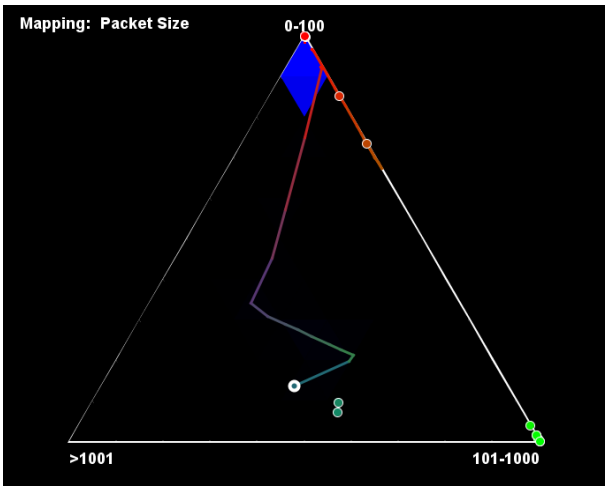

Figure 7: The Tri-Linear Visualization mapped to packet size.

Another interesting discovery in this example is the node in the center that has moved from the top vertex down to the lower middle of the plot. The node is selected, and so by the zones it is clear to see that the node had spent a great deal of time sending only small packets, and suddenly, it began sending much larger packets. The connection it represents has clearly changed the type of data it is sending from small single packets to larger packets. This is likely the beginning of a file transfer that will produce bigger full Ethernet packets.

A third and final example shown in Figure 8 shows a network capture from a single computer. The single computer was accessing the Internet to watch a streaming video. After running only the streaming video, another single web page was opened, producing a collection of other nodes. Upon detailed investigation, it is clear that the collection of nodes are coming from the original page, as well as other sites that are providing additional content, like images, ads, and so on. The two purplish nodes heading towards the lower left corners have likely begun transmitting larger files, like images or Flash objects.

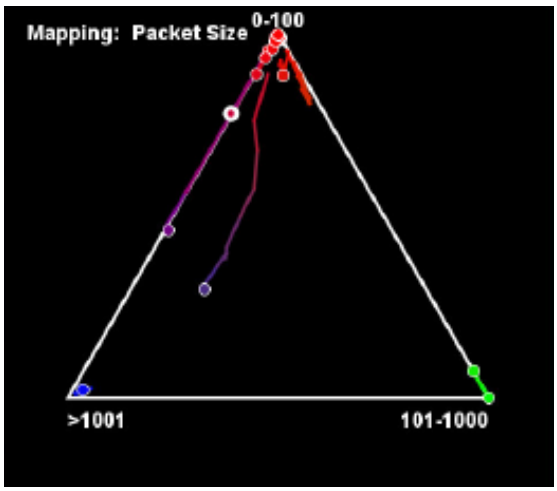

Figure 8: The Tri-Linear Visualization with video and web traffic.

6 Conclusions and Future Work

The results show that the Tri-Linear Visualization is capable of detecting a variety of network attacks and significant events. With continued development, the visualization could prove to be even more powerful. We have seen that using anomaly-based visualization can alert the user to when something is amiss on the network. We have shown that port scans are easily visible in the visualization (including low and slow scans), as are sudden changes in the behavior of a computer. These events allow a network analyst to quickly see what may be wrong on the network.

One of the best features of the Tri-Linear Visualization is that it has proven to be easy to use. Many other visualizations are three dimensional, which requires the user to rotate the display to understand what is going on. A simple 2D display like the one used in the Tri-Linear Visualization allows a user to be able to analyze the network without having to work with complicated user interactions to control an abstract 3D environment. A user can tell what might possibly be wrong by simply looking at things that are changing in the display, especially the movement of nodes. “Normal” nodes stay put, while others move around.

The Tri-Linear Visualization has one significant limitation that will most likely not be able to be fixed. The fact that all pieces of data must be categorized into one of exactly three categories is the driving force behind the underlying ternary plot, but it is also an intrinsic limitation to the plot and to the visualization as a whole.

There is another limitation that the Tri-Linear Visualization has that is common to all anomaly-based systems; namely, a system can perform bad activity so often that malicious events become normal. This is not likely to occur in a real world situation, however, and for now, we are ignoring this concern.

Scalability also appears to be a significant issue with this visualization technique. Like many other visualizations, a large data set greatly clutters the screen, which restricts what the user can do with it. Since most nodes will have little or no movement, anomalous nodes will stand out easily. By ensuring that node trails are drawn last, they will appear on top of static nodes and make the anomalies clear even in large complicated networks. This visualization is also amicable to other avenues for improving scalability. For instance, since the entire visualization is based on anomalous behavior, a metric of anomalous behavior in a node could be provided. It would also be beneficial to add a transparency component to each node, so that nodes that are behaving within normal bounds are completely transparent, and nodes that are an anomaly appear opaque to varying degrees, depending on how anomalous it is. This could be specifically implemented through identification of nodes with a movement below a specified threshold over a period of time being deemed normal and made transparent. This would ensure that only anomalous data is displayed, and would greatly reduce clutter on the screen. A feature such as this is going to be almost mandatory for this visualization before it could be considered viable for commercial uses.

Another possible improvement to the interface would be to enable the size of nodes to increase and decrease, based on the total amount of data being processed at the given moment. The Tri-Linear Visualization currently analyzes the type of data each node is processing, and almost completely ignores the volume of data. Adding in this feature would greatly

improve its power. If this feature were implemented, though, it would be important for it to show the *change* in volume of data, rather than the actual amount of data, in order to allow the visualization to focus on anomalous behavior. For example, a popular web server like google.com handles a vast number of requests every second, while a single desktop computer only handles a very small amount of traffic. It would be important for this feature to show that Google's server has suddenly begun transmitting an abnormally small amount of data, or that an individual's desktop computer is suddenly being flooded with data, instead of the normal slow trickle. Thus, the size of the node must be based on how abnormal the volume of data is, rather than a measurement of the total volume.

7 References

[1] Anderson J.E. and Schwager P.H., "Security in the Information Systems Curriculum: Identification & Status of Relevant Issues," *Journal of Computer Information Systems*, 32:3, 2002, pp. 16-24.

[2] Abdullah, K., Lee, C., Conti, G., Copeland, J.A., and Stasko, j., "IDS RainStorm: Visualizing IDS Alarms," *in Proceedings of Visualization for Computer Security '05*, 2005, pp. 1-10.

[3] Ball, R., Fink, G., and North, C., "Home-Centric Visualization of Network Traffic for Security Administration," *Proceedings of the 2004 ACM Worksop of Visualization and Data Mining for Computer Security*, 2004, pp. 55-64.

[4] Depren, O., Topallar, M., Anarim, E., and Ciliz, M., "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Systems with Applications*, Vol. 29, No. 4, 2005, pp. 713-722.

[5] Erbacher R.F. and Garber M., "Real-Time Interactive Visual Port Monitoring and Analysis," *Proceedings of the International Conference on Security and Management*, June 2005, pp. 228-234.

[6] Erbacher, R.F., Christensen, K., and Sundberg, A., "Designing Visualization Capabilities for IDS Challenges," *Proceedings of the 2005 VizSec Workshop*, Minneapolis, MN, October 2005.

[7] Fink, G.A., Muessig, P., and North, C., "Visual Correlation of Host Processes and Network Traffic," *in Proceedings of Visualization for Computer Security '05*, 2005, pp. 11-20.

[8] Fischer, F., Mansmann, F., Keim, D.A., Pietzko, S., and Waldvogel, M., "Large-scale Network Monitoring for Visual Analysis of Attacks," *Proceedings of the 5th International Workshop on Visualization for Computer Security*, Lecture Notes in Computer Science, Vol. 5210, 2008, pp. 111-118.

[9] Irwin, B., and van Riel, J.P., "An Interactive Attack Graph Cascade and Reachability Display," *Proceedings of the 5th International Workshop on Visualization for Computer Security*, Springer, 2008, pp. 221-236.

[10] Kabiri, P. and Ghorbani A., "Research on Intrusion Detection and Response: A Survey," *International Journal on Network Security*, Vol. 1, No. 2, 2005, pp. 84-102.

[11] Kruegel, C., Toth, T., Kirda, E., "Service specific anomaly detection for network intrusion detection," *Proceeings of the 2002 ACM Symposium on Applied Comupting*, 2002, pp. 201-208.

[12] Lakkaraju, K., Lee, A.J., and Yurcik, W., "Nvisionip: netflow visualizations of system state for security situational awareness," *In Proceedings of CCS Workshop on Visualization and Data Mining for Computer Security, ACM Conference on Computer and Communications Security*, October 29, 2004.

[13] Livnat, Y., Agutter, J., Moon, S. Erbacher, R.F., and Foresti, S., "A Visualization Paradigm for Network Intrusion Detection," *Proceedings of the IEEE Systems, Man and Cybernetics Information Assurance Workshop*, June 2005, pp. 92-99.

[14] McPherson, J., Ma, K., Krystosek, P., Bartoletti, T., and Christensen, M., "PortVis: A Tool for Port- Based Detection of Security Events," *Proceedings of CCS Workshop on Visualization and Data Mining for Computer Security, ACM Conference on Computer and Communications Security*, October 29, 2004.

[15] Mertie, J., "Transformation of Tri-linear and Quadriplanar Coordinates to and from Cartesian Coordinates," *The American Minerologist*, Vol. 49, No. 7/8, July-August, 1964, pp. 926-936.

[16] Suo, X., Zhu, Y., and Owen, S., "A Task Centered Framework for computer Security Data Visualization," *Proceedings of the 5th International Workshop on Visualization for Computer Security*, Lecture Notes in Computer Science, Vol. 5210, 2008, pp. 87-94.

[17] United States Department of Agriculture Natural Resources Conservation Service, *Soil Survey Manual*, 12 November 2008.