LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-217880

A Visualization Methodology for Characterization of Network Scans

C. W. Muelder, K. Ma, A. Bartoletti

December 27, 2005

Visualization for Computer Security 2005
Minneapolis, MN, United States
October 26, 2005 through October 26, 2005

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

A Visualization Methodology for Characterization of Network Scans

Chris Muelder*
University of California, Davis

Kwan-Liu Ma†
University of California, Davis

Tony Bartoletti‡
Lawrence Livermore National Laboratory

ABSTRACT

Many methods have been developed for monitoring network traffic, both using visualization and statistics. Most of these methods focus on the detection of suspicious or malicious activities. But what they often fail to do refine and exercise measures that contribute to the characterization of such activities and their sources, once they are detected. In particular, many tools exist that detect network scans or visualize them at a high level, but not very many tools exist that are capable of categorizing and analyzing network scans. This paper presents a means of facilitating the process of characterization by using visualization and statistics techniques to analyze the patterns found in the timing of network scans through a method of continuous improvement in measures that serve to separate the components of interest in the characterization so the user can control separately for the effects of attack tool employed, performance characteristics of the attack platform, and the effects of network routing in the arrival patterns of hostile probes. The end result is a system that allows large numbers of network scans to be rapidly compared and subsequently identified.

Keywords: information visualization, security visualization, graph visualization, clustering, wavelets, scalograms, network scans, cyber forensics, adversary characterization

1 INTRODUCTION

Scanning a network is a very common first step in a network intrusion attempt. The process of scanning a network is usually performed in order to determine what exists on a network. For example, if an attacker is looking for web servers, then he or she would attempt to connect on TCP/UDP port 80 to every possible IP address within a certain range. For each of these attempted connections, if there is a web server using port 80 with that IP address, it will probably respond. However, if there is nothing at that address, or if there is a computer that is not running a web server, there will be no response. There are scans going on continuously on the Internet, ranging from productive activity such as web crawlers, to malicious activity such as worms [14]. Being able to differentiate between such activity is one of the reasons that it is desirable to be able to categorize and identify the source of a scan. Many visualization tools exist to detect scans [4, 3], but they are often incapable of performing any sort of classification.

An attacker can do several things to attempt to make such a scan anonymous, such as coming from different source addresses or scanning destination addresses in a random order. In fact, it is even possible to perform a scan indirectly, by using a fake source address so the scan looks like it is coming from a different computer. Denial of service and worm propagation attacks can also produce scan like behavior, and since they do not need the target to respond, they often fake their source addresses as well. The port number is also not sufficient for categorizing scans, because a malicious scan can often be run on the same port number as a benign one. For example, both a web crawler and a worm that targets webservers would both target port 80. Therefore, some other metric must be used for categorization purposes.

Variations in arrival time of the scanning connections can be used to classify such an attacker. However, the chaotic nature of these variations makes direct comparison both unwieldy and unreliable. What were needed were measures sufficient to identify characteristic patterns despite pointwise distortions and differences. In order to accomplish this, some visualization concepts have been developed for viewing individual scans, and some statistical concepts have been developed that are useful for comparisons between scans. What is discussed here is a means to combine these concepts along with a graph visualization method to allow for rapid comparison and identification of large numbers of network scans.

1.1 Related Work

The work presented in this paper has evolved from the scan characterization work of Bryan Parno and Tony Bartoletti [12], but there are several other systems that share some similarities. The concept of using visualization in order to characterize attacks, scans in particular, has been explored in [2], which uses a parallel coordinates system to display scan details. *Mirage* [6] is a modular visualization tool designed to handle generic comparisons of multidimensional data. It utilizes many visualization techniques such as clustering, parallel coordinates, and scatter plots to display trends in the data to the user. Each of these techniques is encapsulated in its own panel, and these panels can be added, removed, and configured dynamically by the user. Because of its generality, it could easily be used to analyze scans by comparing scan fingerprints. *PortVis* [10] uses a drill-down approach to display hourly network traffic statistics summarized by port for datasets as large as entire weeks. It uses a grid-based method of displaying values for each port, where each point in a grid is colored based on some metric. Since the data it deals with is at a high level of abstraction, it is good at detecting anomalous activity, but not at investigating the details of such activity. *NVisionIP* [8] and *SeeNet* [1] also use grid-based visualization, where each axis of the grid corresponds to network addresses, and the color of each point in the grid corresponds to a network statistic for the addresses corresponding to its two indices. Then, activity such as network scans show up as lines in this plot, which are easily recognizable. *The Spinning Cube of Potential Doom* [9] extends the grid-based visualization techniques to a three dimensional volume, where the axes represent source ip, destination ip, and port. Then, port scans and network scans can both show up as distinctive lines, but in different directions. Several of the techniques used in these systems are also in our methods, such as grid based visualizations, graph based visualizations, and a combination of an overview with a detailed view. But what differentiates our method from prior systems such as these is a cyclic interaction between the overview and the detailed view.

*muelder@cs.ucdavis.edu
†ma@cs.ucdavis.edu
‡azb@llnl.gov

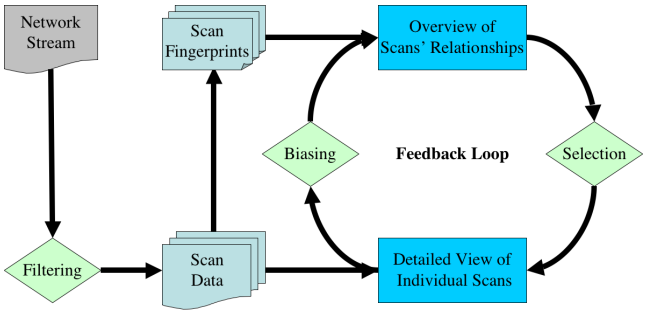


Figure 1: The whole process, from data collection to display with a feedback loop. Raw network data gets filtered into isolated scans, which are then used to create sets of fingerprints The fingerprints are used to create an overview, from which scans can be selected to be viewed in detail. Then, user input can be fed back into the overview for future reference.

2 Methodology

Visualization often tends to be a cyclical process, where each iteration provides more insight into the data being shown. A typical example of this process occurs with any type of overview plus detail visualization. Patterns in the overview tend to direct what the user chooses to view in more detail, and the detailed view can provide insight on regions of the overview, and direct how the user cognitively views and manipulates the overview. This process creates a feedback loop which can lend itself well to visualizing the relationships between large numbers of objects, such as network scans.

The methodology that is shown in this paper is based on these concepts as they are applied to viewing large numbers of network scans. Figure 1 shows the entire process from data collection to the visualization feedback loop. The first step involves filtering the original network stream into isolated network scans, in order to visualize them. At this point, the scans can be directly visualized individually, but when dealing with large numbers of scans, this is unfeasible. So, once the scans are isolated, in order to automatically compare them, fingerprints are generated to be fed into an overview visualization. This overview of the relationships between the scans and the detailed view of individual pairs of scans for comparison purposes compose an overview plus detail feedback loop. As described before, the overview allows the user to drill down into certain areas, by showing them in the detailed view. However, unlike most overview plus detail visualizations, ours allows the user to bias the overview in a manner reflecting the cognitive insight gained from looking at the detail view. This enhances the feedback loop by allowing information gained by viewing the details of network scans to be reflected back in the overview.

2.1 Data Filtering

When dealing with large networks, such as a class B network with nearly 65536 possible addresses, the process of scanning the address space is not feasibly doable by hand, and unless it is done rapidly, it can take a long time to complete. Thus, many scans are performed quickly and noisily using an automated tool or script. This kind of activity can be easily and automatically detected by monitoring a network stream for connections to many destinations in a small space of time. When a certain threshold of destinations per unit time is detected, it is usually a trivial task to extrapolate forwards and backwards in time to extract the entire scan. Of course, it is possible to miss a scan if it is being performed at a rate lower than the threshold, and at the other extreme, it is possible to categorize normal traffic as a scan if the threshold is too low. However, there has also been work done in the area of using visualization to detect patterns such as network scans in the network traffic, such as PortVis [10]. Once a scan is detected with a visualization tool or even a different statistical method, it can be extrapolated and extracted from the network stream in the same manner as the ones detected with a threshold.

2.2 Scan Data

This extraction process creates a set of individual network scans, consisting of a single source, a destination port, and pairs of destination addresses and packet arrival times. Once these scans are isolated, it is desirable to compare and contrast them, so that the sources can be categorized by factors other than IP addresses or port numbers. While IP addresses and port numbers can be helpful for categorization purposes, IP addresses can be spoofed and malicious scans can be run on the same ports as benign ones. It was determined experimentally that looking at the packet timing and sequencing information could provide good metrics for doing this, because timing and ordering can be influenced in different ways by many factors which are directly related to classification. Some of these factors that can alter the arrival time of a packet are the tool or tools used to generate the scan, the attacker's hardware, the attacker's operating system, and even router delays. Therefore, each scan has been reduced to a collection of pairs of destination addresses and times. Not every scan hits the entire network space, and some scans hit destinations more than once, so there is not necessarily exactly one time for any given destination.

2.3 Scan Fingerprints

Direct comparisons of these datasets is not always useful because similar patterns from the same source could even be completely different for every value. For example, say an attacker scans every other destination of a network, then a day later scans the other destination addresses that were skipped in the first scan. The scans would exhibit nearly identical traits, but a direct comparison would say that they were completely different. Therefore, some more complicated form of pattern matching must be performed, such as comparing fingerprints of the scan. Since fingerprints are often much smaller then the original data, this can have the added effect of reducing the size of the data being compared by an order of magnitude. However, one has to be cautious in selecting the function used to generate such fingerprints, or else the distinguishing features of the patterns being observed will be lost in the reduction process.

2.4 Feedback Loop

Many cyclic visualization methods require the user to remember insights learned from low level semantic views of the data in order to aid interpretation of higher level semantic views. In order to improve upon this cyclic process, the capability was added to allow the user to record cognitive insights gained from the detail view into the set of relationships shown in the overview. This allows the user to feed knowledge back into the system, which is useful for keeping track of large numbers of cognitive insights. Repeated iterations of this feedback loop will refine the overview of the data, making it easier to understand. And by saving this refined relational data, future datasets containing the same scans or different views of the same dataset can recall the user defined feedback, allowing the user to more rapidly understand the data.

3 A Scan Visualization System

The two primary visualizations needed to implement this methodology are an overview showing the relationships between network

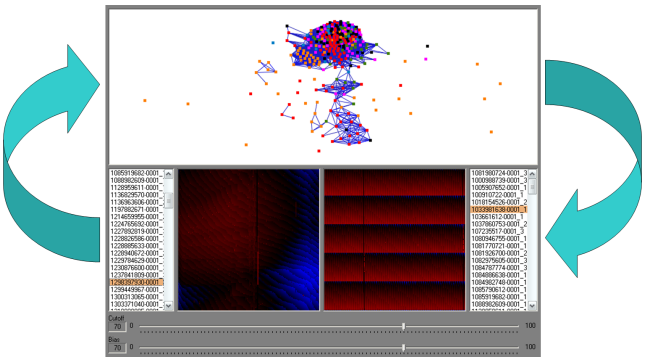


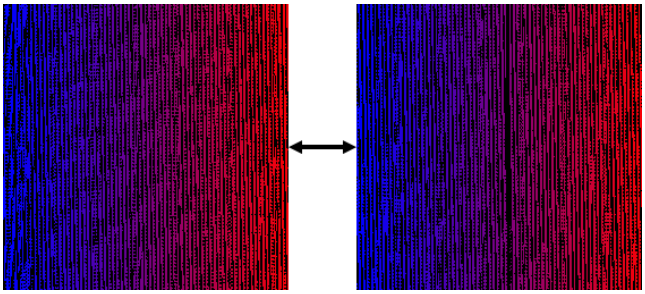
Figure 2: *The interface of the scan visualization system*: Both the high level and the low level views are presented in a single screen so the user can go back and forth easily.

scans, and a detailed view showing individual scans. Since the final goal for the overview is to visualize the relationships between network scans, it was determined that showing these relationships directly in a graph visualization would be somewhat intuitive. Then the relationships can be shown spatially using a force directed algorithm to place similar scans next to each other. This requires that the scans be compared statistically in order to automatically determine the strength of the relationship between any two scans. Also, for the detailed view, there are many different metrics based on timing that can be shown, but the scans are of a fixed size network, consisting of 65536 consecutive IP addresses. So it makes sense to use a grid based visualization of fixed size and let the color at each point in the grid represent the currently desired metric. In order to implement these visualization concepts, a system was developed in C++ using wxWidgets and OpenGL. As is shown in Figure 2, it presents both the high level overview and the low level detailed view simultaneously, so that the user can go back and forth between them rapidly. The user can select scans from the overview to be viewed in pairs in the detail view. And the user can use a standard slider to alter the numerical measure of the similarity between scans, which is then reflected in the overview graph. Thus, it presents an interactive view of the data, allowing the user to iteratively refine an understanding of the data.

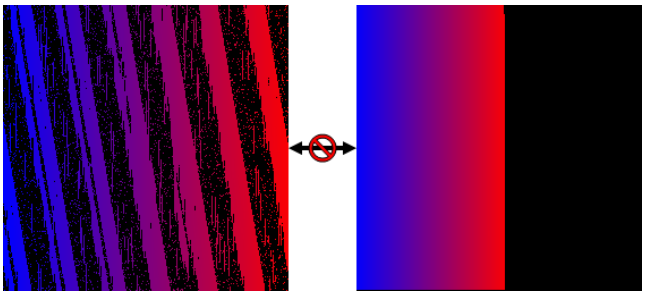
3.1 Data Modes

The raw scan data is composed of arbitrary pairs of destination addresses and times. In order to generate a more useful visualization, various transformations were performed to show different aspects of the data. This created a set of derived metrics which can be shown in the detail visualization. For example, one simple metric is to record the time of the first connection attempt to each address. Some more complex metrics are defined as follows:

- m20: f(a) = N(v), the number of visits per unique address
- m21: f(a) = tFirst - tLast, the revisit-span for each address
- m22: f(a) = tFirst - E(tFirst), time deviance for first probes
- m23: f(a) = tLast - E(tLast), time deviance for last probes
- m24: f(a) = d(tFirst), time delta on sequential addresses, first probe
- m25: f(a) = d(tLast), time delta on sequential addresses, last probe



(a) Similar scans have similar patterns



(b) Dissimilar scans have dissimilar patterns

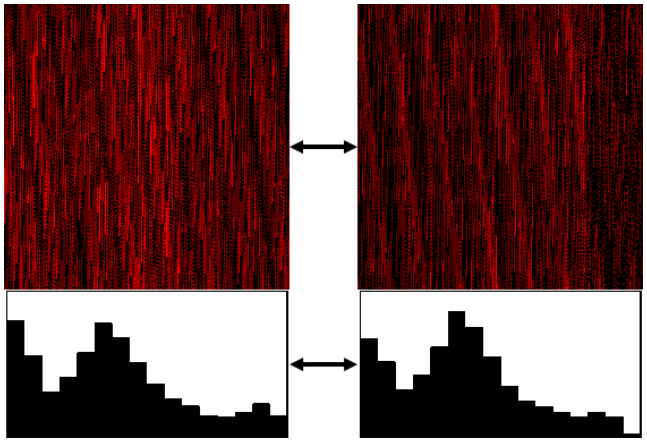
Figure 3: Visualization of individual scans shows patterns that can easily be compared by eye. These particular images show the arrival time of the first connection attempt to each address, with blue being early in the scan, red being late in the scan, and black being an address that had no connection attempt.

3.2 Displaying Individual Scans

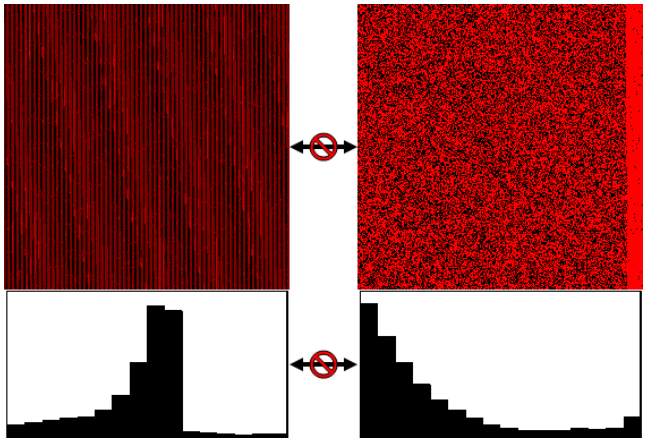
One half of the feedback loop is a detailed view technique for displaying a single scan. One such technique is displaying the scan in a 256x256 grid, where the x and y coordinates are the third and fourth bytes of the destination IP addresses scanned ("c" and "d" in the address a.b.c.d), and the color represents various metrics based on statistical information regarding the arrival time at that IP address. This allows an entire class B network to be shown in a single image. Many existing visualization tools use similar methods, such as PortVis [10] and NVisionIP [8], and some even extend this concept to three dimensions, such as the Spinning Cube of Potential Doom [9]. As can be seen in Figure 3, some scans exhibit similar patterns, while others exhibit quite different patterns when displayed in this manner. The color gradient used in this image is simply a blue to red gradient, where blue is early in the scan, red is late, and black was an address with no activity. The other color gradient used is a blue to black to red, where blue indicates a negative value, black indicates a zero value, and red indicates a positive value. This tends to work well when there are negative values, as in the deviation from expected time metric. Another possible technique is to plot ip versus arrival time and contrast it to the line that corresponds to the linear expected arrival times, but this does not tend to scale very nicely to the large number of connections in any given scan.

3.3 Side by Side Comparisons

Being able to compare two patterns is one of the primary tasks that are very useful to the process of pattern matching. In order to simplify this task as it relates to network scan characterization, two 256x256 panels were used so that scans could be compared side by side. When any scan is selected in the overview, it and its neighbors are placed in a pair of lists. Each of these lists has a corresponding



(a) Similar scans have similar wavelet scalograms



(b) Dissimilar scans have dissimilar wavelet scalograms

Figure 4: Wavelet scalograms reduce large complex patterns to smaller simpler vectors that can be compared. This example was made using data mode m20, with a black to red gradient, where black is no probes and red is the maximum for that scan.

256x256 scan display panel, and scans can be selected from either list, and displayed in the respective scan display panels. In each panel, scans are displayed using the 256x256 grid-based visualization techniques. Then, the user can view them and if the fingerprint comparison process is not an accurate representation of the similarity of the two scans, the user can then bias the weight corresponding edge in the graph. This could occur when the wavelet analysis missed some feature difference between the two, so they need to be biased negatively, or when the wavelets amplify minute differences, so they get labeled as being different, even though they are quite similar. This process completes the feedback loop back into the overview graph, which allows the user to progressively refine the accuracy of the data representation.

3.4 Wavelet Scalograms

The first step in generating an overview of the relationships between network scans is to statistically compare pairs of scans and get a quantitative measure on how well they match. The scans are too chaotic to easily directly compare, but there are several algorithms utilizing frequency analysis that are useful for handling this kind of data, such as Fourier transforms and wavelet scalograms. Although network scan patterns can exhibit periodic or quasi-periodic structure, they often contain gaps, aperiodic aberrations and regions where the relative phase of the periodic structures has shifted, which are things that Fourier analysis has been found to not handle well [5].

So, wavelets are used because they are relatively resistant to phase shifts and noise. That is, similar patterns will have similar wavelet scalograms, even if the patterns are shifted slightly or different parts of the pattern are missing. However, dissimilar patterns should produce different scalograms, as can be seen in Figure 4. There are several variations on the wavelets used, the simplest of which is as follows. Given a series of $N = 2^n$ items $D_0 = (d_{0,1}, d_{0,2}, ..., d_{0,N})$, we can calculate recursively:

- $D_k = (d_{k,1}, d_{k,2}, ...d_{k,2^{n-k}})$
  $= (\frac{d_{k-1,1}+d_{k-1,2}}{2}, ..., \frac{d_{k-1,2^{n-k}-1}+d_{k-1,2^{n-k}}}{2})$
- $S_k = (s_{k,1}, s_{k,2}, ...s_{k,2^{n-k}})$
  $= (\frac{|d_{k-1,1}-d_{k-1,2}|}{2}, ..., \frac{|d_{k-1,2^{n-k}-1}-d_{k-1,2^{n-k}}|}{2})$
- $\sigma_k = \sum \frac{S_k}{2^{n-k}}$

for $0 < k < n$. At each recursion the σ values are the mean of the corresponding data series, which estimates the variance at each resolution. More complicated wavelets can be calculated by changing the functions used to calculate D_k and S_k. Some of the functions being used currently are:

- w00:
  $d_{k,i} = \frac{|d_{k-1,i}+d_{k-1,i+1}|}{2}$
  $s_{k,i} = \frac{|d_{k-1,i}-d_{k-1,i+1}|}{2}$
- w03: (a.k.a. Haar wavelet)
  $d_{k,i} = \frac{d_{k-1,i}+d_{k-1,i+1}}{\sqrt{2}}$
  $s_{k,i} = \frac{d_{k-1,i}-d_{k-1,i+1}}{\sqrt{2}}$
- w04: (mean of 0 and $\frac{1}{4}$ phases)
  $d_{k,i} = \frac{|d_{k-1,i}+d_{k-1,i+1}|}{2}$
  $s_{k,i} = \frac{|d_{k-1,i}-d_{k-1,i+1}|}{2}$
- w10: (a.k.a. "pointwise" wavelet)
  $d_{k,i} = \oslash$ (acts directly on D_0)
  $s_{k,i} = \frac{\Sigma_{j=0}^{2^{k-1}} |(d_{0,i+j}-d_{0,i+2^{k-1}+j})|}{\sqrt{2}}$
- w23:
  $d_{k,i} = \sqrt{d_{k-1,i}^2 + d_{k-1,i+1}^2}$
  $s_{k,i} = Tan^{-1}(\frac{d_{k-1,i+1}}{d_{k-1,i}})$
- w24: (a.k.a. "Absolute Slope Product")
  $d_{k,i} = \frac{|d_{k-1,i}+d_{k-1,i+1}|}{2}$
  $s_{k,i} = |d_{k-1,i+1} - d_{k-1,i}| * |d_{k-1,i+3} - d_{k-1,i+2}|$
- w29: (a.k.a. "Slope Sign")
  $d_{k,i} = \frac{d_{k-1,i}+d_{k-1,i+1}}{2}$
  $s_{k,i} = \begin{cases} -1 & \text{if } d_{k-1,i} > d_{k-1,i+1} \\ 1 & \text{if } d_{k-1,i} < d_{k-1,i+1} \\ 0 & \text{if } d_{k-1,i} = d_{k-1,i+1} \end{cases}$

Averaging with the quarter phase function, as in w04, adds complexity, but can be useful when the other methods would miss something, or classify something incorrectly. For example, a square wave could show up as a spike in a completely different frequency if the phase is off. But with the quarter phase averaging, a similar but more consistent pattern is shown even when the phase changes.

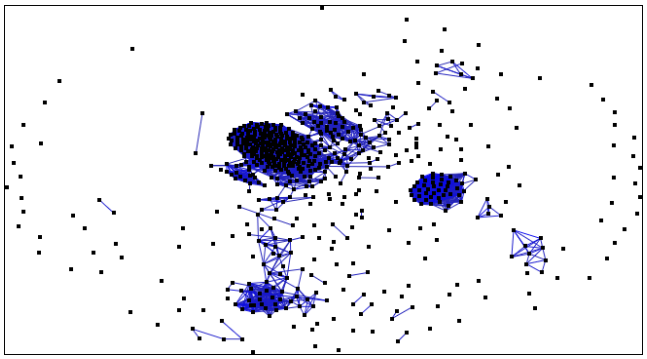


Figure 5: A graph of 681 nodes, showing clusters. Generated from data mode m20 and wavelet w04, with a geometric mean comparison and a 70% cutoff. The force directed layout causes scans with similar scalograms to end up next to each other.

3.5 Wavelet Comparisons

The next challenge is taking the multidimensional wavelet signatures and measuring how well they match with a single number to be used as an edge weight. This can be done in many different ways, such as taking the arithmetic mean of the relative differences, the geometric mean of the differences, or the inverse of the Euclidean distance between the two signatures. The methods currently used are calculated as follows. Let $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$ be the scalograms corresponding to two nodes, then the weight w of the connection between them is:

- Arithmetic: $w = \sum_{i=1}^{n} \frac{1 - \frac{|a_i - b_i|}{a_i + b_i}}{n}$
- Geometric: $w = \sqrt[n]{\prod_{i=1}^{n} (1 - \frac{|a_i - b_i|}{a_i + b_i})}$
- Geometric(no root): $w = \prod_{i=1}^{n} (1 - \frac{|a_i - b_i|}{a_i + b_i})$
- Inverse Euclidean: $w = \frac{1}{1 + \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}}$
- Inverse Euclidean^2: $w = \frac{1}{1 + \sum_{i=1}^{n} (a_i - b_i)^2}$

Different methods can be more or less effective with different wavelet and data mode combinations. For example, the arithmetic and geometric means are calculated relatively, so they are more effective when there are large changes in the wavelets relative to the average value. However, if the wavelets tend to have large average values, and the relative changes are small, then one of the Euclidean functions would likely work better.

3.6 Overview Graph of Scan Relationships

A graph visualization was developed in order to provide a high level view of a large set of scans. In the graph, each node represents a scan, and the connection between any two nodes is weighted according to the wavelet comparison between them, with 0% being completely different and 100% being identical. Since this is a complete graph, a cutoff was added to simplify the graph – any connection where the nodes match less than some threshold is dropped. Then by using the LinLog force directed layout [11], nodes with higher match percentages attract each other more than nodes that do not match as well. So nodes that correspond to the same source cluster together. This effect can be clearly seen in Figure 5, which shows a graph of 681 scans. Additional information can also be shown in the color of the nodes. Any given node can be colored according to destination port, mean scan rate, elapsed time, or number of probes. While these do not directly affect the clustering, often after the clusters have formed, patterns can be seen in the data, such as clusters that are all one value or that have a smooth gradient in one metric.

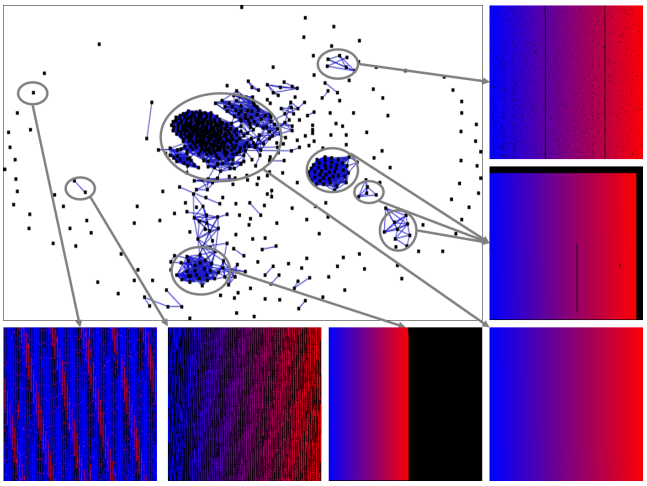


Figure 6: Clusters contain scans with a general pattern. A representative example from each selected cluster is shown.

4 Case Studies

Much can be learned from simply drilling down into the clusters that are generated automatically. For example, in the graph shown in Figure 6, there are several clusters, both large and small, that have distinctive characteristics which can easily be seen in the representative examples shown. Starting with the one in the upper right of the figure, the first cluster's representative example shows a mostly continuous scan from low numbered to high numbered IP addresses, but with some speckled areas where some addresses were missed. Continuing clockwise, the next example is a representative of three separate clusters that showed a similar pattern under this data mode, which is a region around the top and the right of the grid display in which there were no connection attempts. The next cluster is the single largest cluster, and it corresponds to a simple continuous scan of nearly the entire subnet. However, there are still occasional addresses to which no connection was detected. The next cluster is of scans which cover only part of the address space, usually about half. However, this cluster extends towards the cluster containing the complete scans, and the closer one gets to this cluster, the larger the region of the subnet that the scan covers. The example second to the left shows a cluster that consists of only two scans, each covering about half of the address space in an interestingly spaced pattern. Also of interest is that both of these scans actually originated from the same source address, hinting that there is a very strong correlation between them. The final representative scan shows one of the many scans that ended up isolated from all the other scans. As can be seen in the representative image, the patterns in these isolated scans can be quite different from the patterns found in clustered scans. This essentially shows how the first half of the feedback loop works, which is the way most overview plus context visualizations work. The user can start with the overview graph and then drill down to detailed views of the scans to gain understanding about the clusters in the graph. The feedback loop can be used to regurgitate this information back to the graph, but this

task can be aided by starting with a wavelet that gives a good initial relationship graph.

Looking at different data modes or different wavelet functions can reveal many interesting patterns. Figure 7 and Figure 8 shows what happens to the graph visualization when different data modes or different wavelets are used for the initial layout. Wavelets w04 and w10 act somwhat similar in their clustering in both data modes: In all four of those combinations, the graph produces one large cluster, and several smaller clusters. These wavelets seem to primarily distinguish nodes mostly by large scale patterns which is indicative of causes such as the tool used to generate the scan. This could be due to the Euclidean distance metric being overwhelmed by large changes at a certain frequency. That is, large scale differences in the pattern are likely inducing large differences in a single value of the scalograms, which is overpowering the smaller changes in other positions of the scalogram that could be caused by factors such as routing delays. For example, when scans in the graph of m20 with w04 are viewed, it can be seen that scans that are near the beginning of the tail are complete, while scans near the end contain only about half the address space. The difference between data modes m20 and m22 when viewed under these wavelets is that data mode m20 shows a smooth gradient between these extremes, while data mode m22 isolates clusters of a particular type better. Wavelet w19 is essentially the same as w04 and w19 for data mode m22, but for data mode m20, it creates a snake-like pattern of clusters, where the ends are completely different, but there are intermediate values in between. This is probably indicative of a particular factor that is being measured, because each scan in the cluster is similar to scans next to it, but not to scans in the same cluster but on different ends. Unlike the previous graphs though, this snake-like pattern is not due to coverage of the address space, since each concentrated cluster contains both scans that cover the whole space and scans that cover only fractions of it. In the combination of wavelet w23 and data mode m20, the resulting graph is somewhat similar to the ones generated by w04 and w10, in that there is one large cluster, one medium sized cluster, and several smaller clusters, and they arrange themselves such that another tail pattern emerges. Also, when the individual scans are viewed, it can be seen that this is caused by the coverage of the address space again. When used with data mode m22 though, wavelet w23 creates a graph where there are multiple snake like clusters, implying that there are groups of scans that cluster together based on unique patterns, but smaller timing factors are probably stretching the clusters out. That is, the overall cluster structure is probably still caused by large differences such as choice of scanning tool, but the position inside the cluster could correlate to timing factors such as hop count. The spread out cluster effect shows up even more strongly, however, in wavelet w24. In both data modes, wavelet 24 creates a pattern where there is a strong cluster in the middle and a trail of nodes that are progressively less similar to the nodes in the middle. When the definition of wavelet w24 is considered, it becomes apparent that this tail-shaped pattern correlates to more or less chaotic data. A large portion of the scans are very straight forward, they start at the first address and proceed to the last address, then possibly repeat this sequence a few times. These scans correspond to the nodes in a large dense portion of the graph, while nodes that do not follow this pattern end up farther away from this point. Of particular interest is how in data mode m22, the stretching of the cluster that wavelet w24 generates corresponds to the scan rate, as can be seen in the gradient of color through the structure. This makes sense, since the faster a scan is run, the more small deviations due to hardware or router delays affect the timing. Wavelet w29 provides the most interesting and unique graphs in both data modes. In mode m20, the wavelet divides the graph quite strongly in two, and then it divides each half more weakly into three. This nearly partitions the graph into 6 separate clusters all of about the same size, although the cluster on the bottom right seems to be close to splitting more. This overall pattern is probably due to the limited number of values encountered in the m20 data mode. Because the data mode consists of very simple data, and the wavelet performs a large amount of simplification, the graph created from their combination exhibits a fairly simple and regular pattern. And finally, the combination of m22 and w29 produces a graph with many separate clusters, which each consist of very similar scans. This implies that this particular combination of data mode and wavelet is categorizing scans according to a wider variety of unique patterns than the other combinations. This is probably because this particular wavelet scalogram function ignores magnitude, so no one frequency can dominate the scalogram. Thus, the large scale frequencies that dominated wavelets such as w04 and w10 are unable to dominate in this w29, so other frequencies can affect the graph more.

Other interesting patterns can be seen just by coloring the nodes differently. In Figure 9, the 5 most scanned ports have been colored according to the given legend. Of interest here is how even though the port was not taken into consideration by the clustering algorithm, there are clusters of almost exclusively port 445 or port 139. For example, the large cluster in the lower right contains the majority of scans on port 139, and no scans on any other port. Similarly, many of the clusters on the left side of the graph are exclusively port 445, so there were several classes of scans running on that port exclusively. This indicates that there were particular kinds of scans that targeted individual ports, which suggests that these scans were not made by a generic scan tool. Had these scans been made by a generic scan tool, there would be scans on different ports with similar timing, such as can be seen in the cluster at the top of the graph. But, each of the clusters of interest contain scans on only a single port, so it is more likely that they were generated by worms, since worms generally run on only one port. The reason the clusters of scans on port 445 do not form a single cluster like the one on port 139 could be due to different variations of a worm, different source systems, different delays over the network, or entirely different worms that just happen to target the same port.

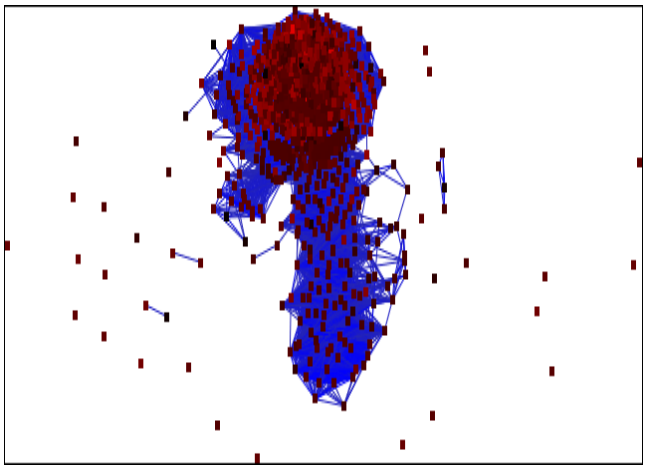
There are many patterns that can be seen with the human eye which even wavelet w29 can not distinguish. This is where allowing the user to bias the edge weights can prove useful. In Figure 10 a single cluster has been isolated out of the graph of data mode m22 and wavelet w29. In it, there were found to be several different patterns that can be easily distinguished by viewing them. So pairs of scans were selected from the graph, viewed side by side, their edge weights biased according to how similar they looked, and the graph was updated according to the new weights. This process was performed on each pair of nodes in the cluster. The end result was that the cluster was split into three separate clusters, each corresponding to one kind of pattern. This shows how one can successfully utilize the feedback loop to refine the graph to give a better representation of the underlying data.

5 Future Work

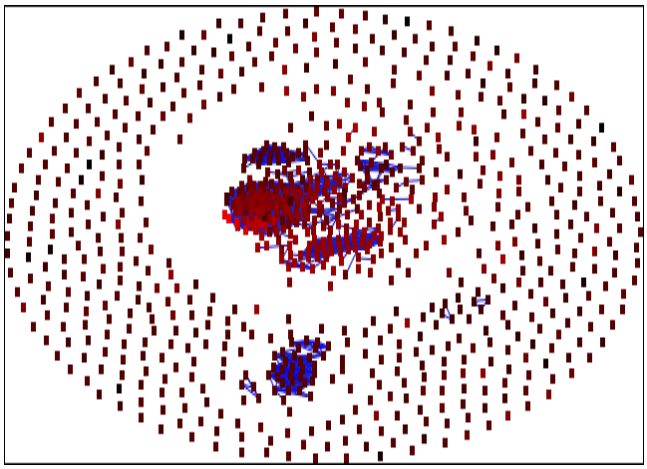
As it is right now, these techniques would likely make a useful tool for categorizing network scans, but there are still several improvements that can be made. There are some other factors that could be useful to visualize besides the wavelet scalograms. It could be beneficial to visualize or at least factor in relationships involving other features, such as the actual source IP addresses or the TCP/UDP ports involved.

5.1 Clustering

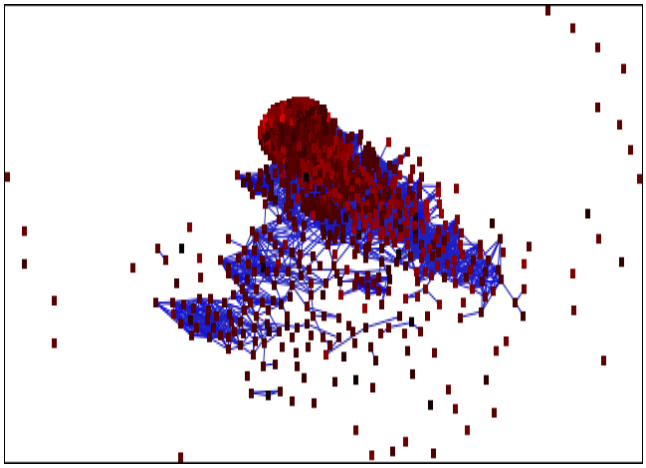
In order to simplify the graph, it would be good to collapse clusters of nodes that match well enough (over some threshold) into a single larger node. Clustering has been shown to be a useful and effective



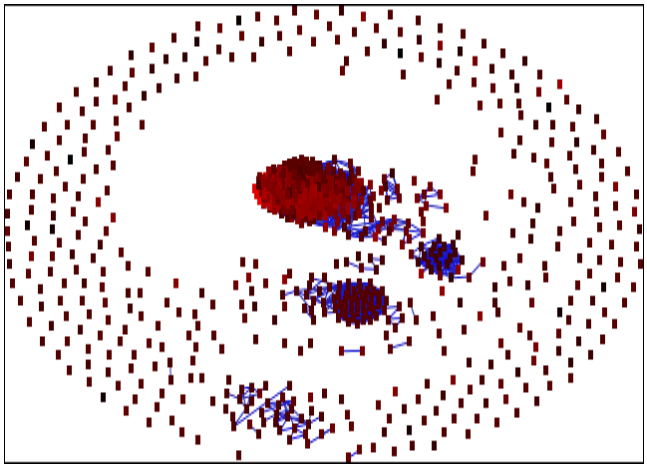
(a) *m20 and w04*: Almost everything is positioned in one cluster, but with a tail leading away from the center due to scans that didnt cover the entire address space.



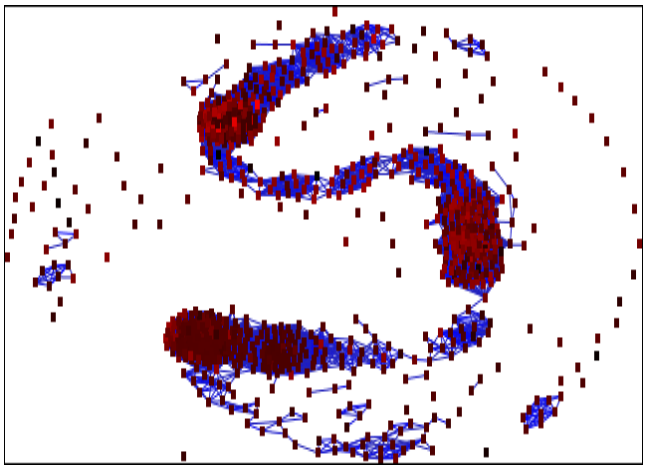
(b) *m22 and w04*: Several clusters of different sizes, but many isolated nodes.



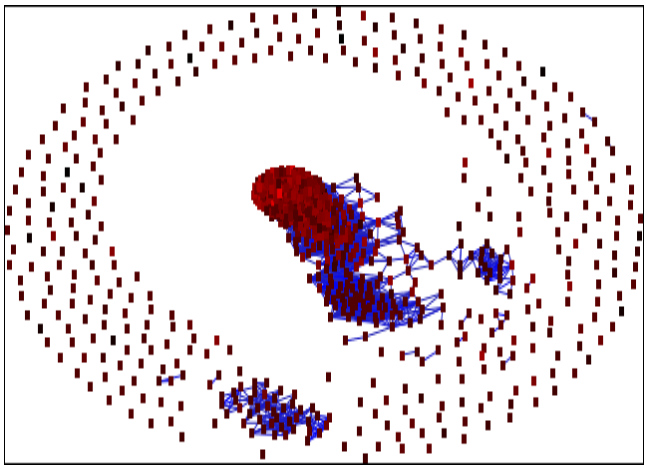
(c) *m20 and w10*: One primary cluster with two tails trailing off.



(d) *m22 and w10*: One large cluster, two smaller clusters, and one weak cluster at the bottom.

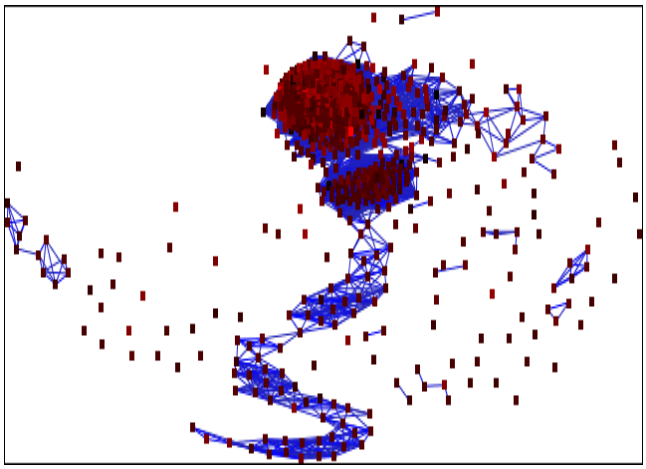


(e) *m20 and w19*: A large snake-like structure running through 4 major concentrated clusters.

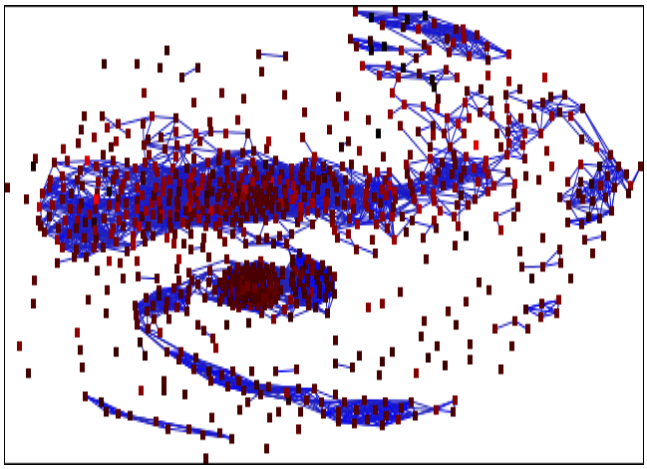


(f) *m22 and w19*: A concentrated cluster with a tail, and 2 other clusters.

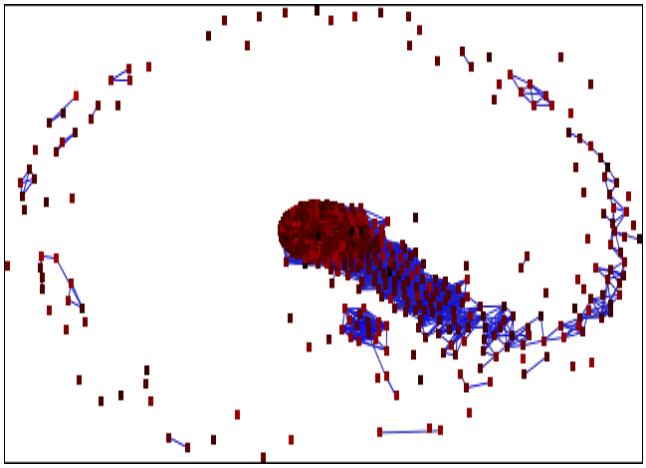
Figure 7: Variations in data mode and wavelet function. Each graph uses an inverse Euclidean squared comparison function, has a 70% cutoff, and consists of 878 scans. Node coloration is based on a log scale of the mean scan rate.



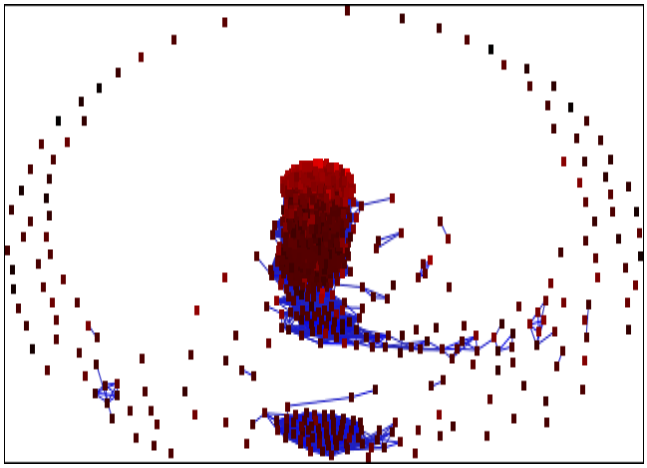
(a) *m20 and w23*: 2 Main clusters, but with a tail running down from them due to address space coverage.



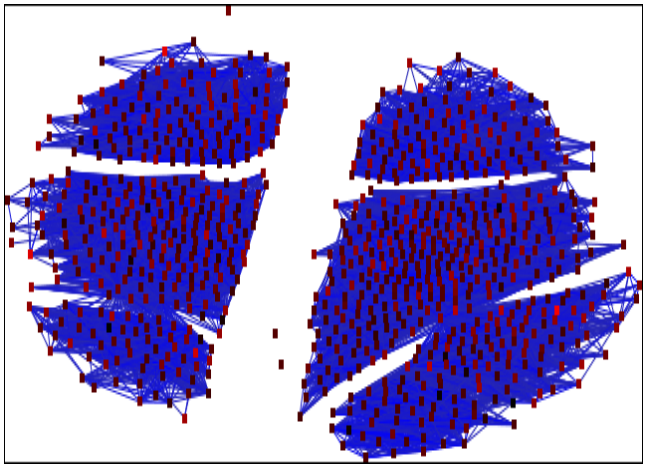
(b) *m22 and w23*: Several clusters, some stretched into snake-like forms.



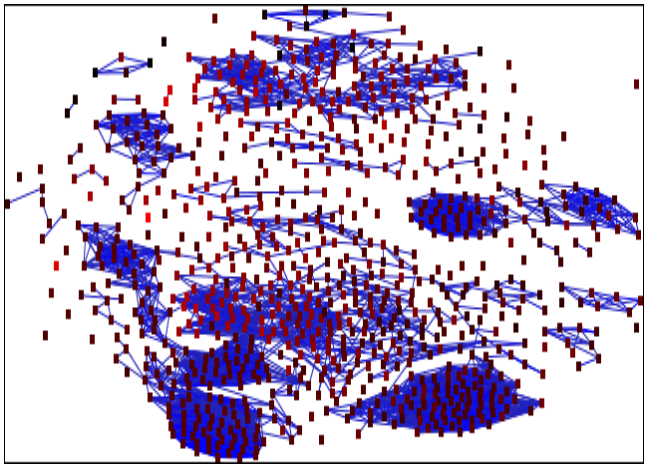
(c) *m20 and w24*: Almost everything forming a snake-like clustering pattern.



(d) *m22 and w24*: A strong snake-like pattern which corresponds to scan rate.



(e) *m20 and w29*: A partition of the graph into 6 regions.



(f) *m22 and w29*: The graph is split into many separate clusters.

Figure 8: More variations in data mode and wavelet function. Each graph uses an inverse Euclidean squared comparison function, has a 70% cutoff, and consists of 878 scans. Node coloration is based on a log scale of the mean scan rate.

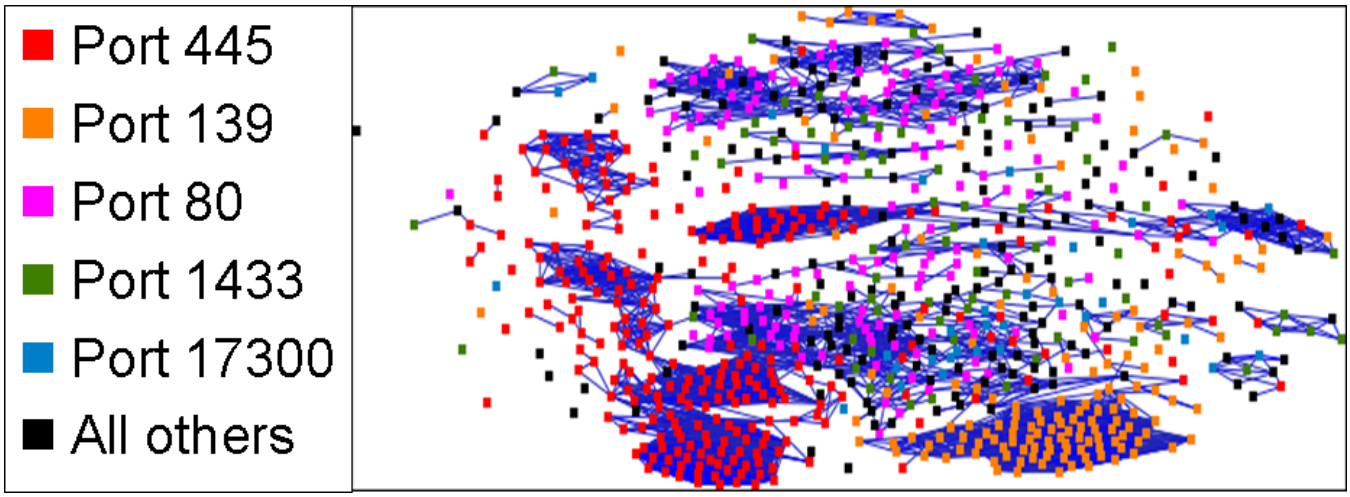


Figure 9: Coloring the nodes based on ports can reveal interesting patterns. In this example, there are several clusters that contain only one port. Namely, the cluster on the bottom right is exclusively port 139, and the the clusters up the left side and the one in the very center are all exclusively port 445. Such patterns are likely caused by worms.

tool in the process of discovering security events in unlabeled data [13], but not much has been done to use it in the process of characterizing such events. Once collapsed, being able to zoom inside of the node to view the internal structure would be useful. However, if the user biases one or more of the internal connections enough that a node or group of nodes falls below the threshold, then the collapsed node would somehow have to be able to split. And vice versa, if the user biases the connection between two separate nodes, they would somehow need to automatically combine into a single collapsed node. The difficulties in this concept lie in choosing an algorithm to collapse closely knit clusters of nodes and choosing how to calculate the relationships between a collapsed node and other nodes for layout and display purposes. One possible algorithm that is being considered is Kohonen's self organizing map algorithm [7].

5.2 Frequency Weighting

Often times, the patterns of interest are focused at a particular frequency level in the wavelet scalograms. Algorithmic choices of the attacker can affect low frequencies quite a lot, but modifying high frequency patterns is nearly impossible algorithmically. Similarly, routing protocols are designed to delay traffic as little as possible, so they would not alter low frequency patterns much, but they would likely create high frequency patterns. Therefore, patterns corresponding to tool or tool flag choices would likely occur at lower resolution scalogram entries, while patterns due to system limitations or Internet routing would more likely occur at higher resolution values in the scalogram. In order to emphasize particular patterns, it would be beneficial to be able to weight different frequencies, in a manner similar to the way equalizers balance audio frequencies.

5.3 Machine Learning

Machine learning could also be easily added. There are many connections in the graph, and if there is a certain feature that the wavelet comparisons show weakly, or miss entirely, the user would have to select and bias all of them, unless some sort of intelligent algorithm does it automatically. However, this is exactly what machine learning algorithms would be good at. As the user biases certain relationships, a machine learning algorithm could be used to bias other similar relationships automatically.

6 Conclusions

The methods described in this paper combine several existing techniques in order to enable the user to categorize and characterize network scans more readily then would be possible using any of the techniques by itself. Utilizing a feedback loop allows the user to correct any inaccuracies created by the wavelet transformations and subsequent wavelet scalogram comparisons. The graph based method, while probably not yet being used to its full potential, provides a relatively intuitive view of the data, even with large datasets. Other methods such as parallel coordinates could work too, but would probably not be as accommodating as the graph based algorithms to some of the operations that it is desirable to perform, as well as being somewhat less intuitive. The process of creating the wavelet scalograms out of the original data loses a lot of detail, as does the process of reducing two multidimensional scalograms to a single edge weight. Therefore, it was desirable to add a feedback loop by allowing users to directly modify edge weights based on looking at more detailed views of the scans. Also, the methods used to categorize these scans do not actually involve anything network specific, so they could also be applied to other fields where pattern matching in regular data series is desirable.

7 Acknowledgments

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE), ACI 0222991, and ANI 0220147 (ITR), ACI 0325934 (ITR), and the U.S. Department of Energy under Lawrence Livermore National Laboratory Agreement No. B537770, No. 548210 and No. 550194. We would like to thank Ellen Raber, Associate Director for the Safety and Environmental Protection Directorate at Lawrence Livermore National Laboratory, for generously supporting this research. We would also thank Andrew Brown and Tim Meier, who have provided network capture support, and Chuck Baldwin for his encouragement and guidance in the exploration of wavelet analysis.

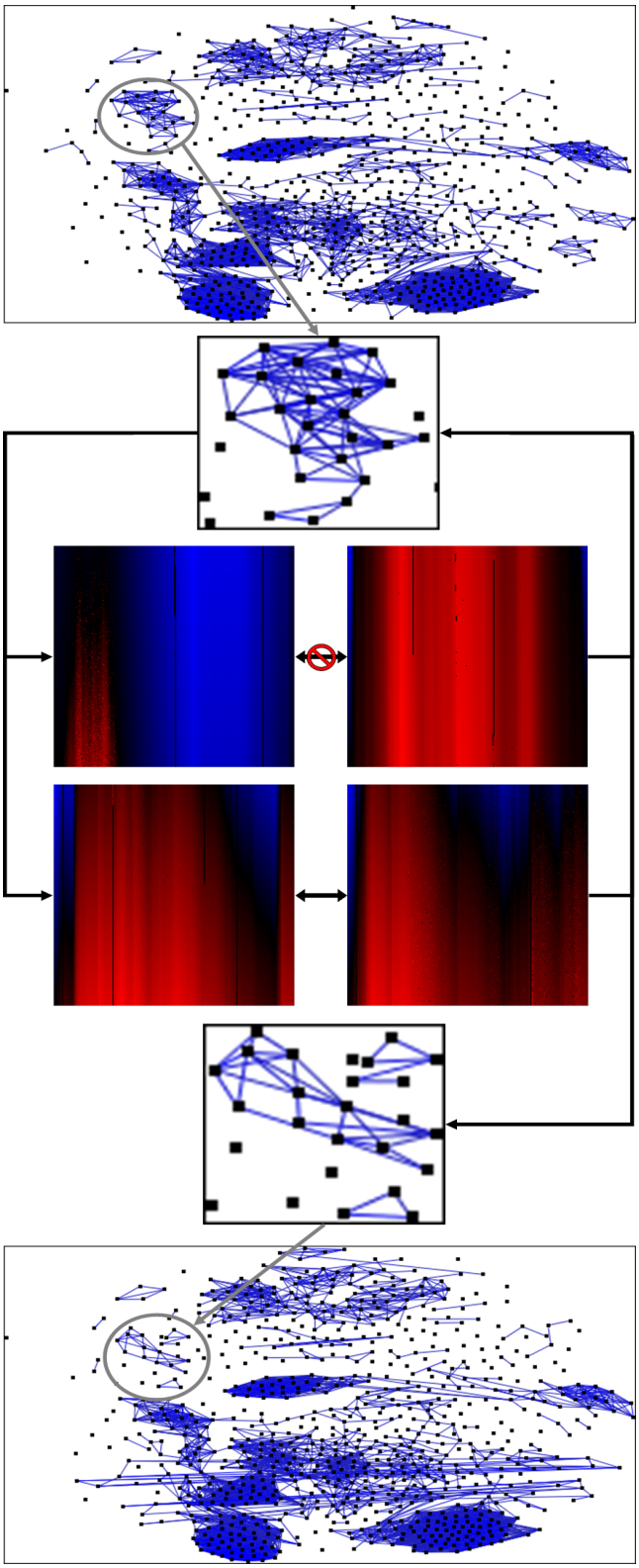


Figure 10: Sometimes clusters are not as homogeneous as one would like. By biasing the weights of the edges, such a cluster can split into several more homogenous clusters. In this example, nodes from one cluster were compared under datamode m22, and their edge weights biased according to how similar they were. This split the single cluster into three separate clusters.

REFERENCES

[1] Richard A. Becker, Stephen G. Eick, and Allan R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, 1995.

[2] Gregory Conti and Kulsoom Abdullah. Passive visual fingerprinting of network attack tools. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 45–54, New York, NY, USA, 2004. ACM Press.

[3] Robert F. Erbacher. Visual traffic monitoring and evaluation. In *Proceedings of the Conference on Internet Performance and Control of Network Systems II*, pages 153–160, 2001.

[4] L. Girardin and D. Brodbeck. A visual approach for monitoring logs. In *Proceedings of the 12th Usenix System Administration conference*, pages 299–308, 1998.

[5] Amara Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, 1995.

[6] Tin Kam Ho. Mirage: A tool for interactive pattern recognition from multimedia data. In *Proc. of Astronomical Data Analysis Software and Systems XII*, 2002.

[7] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3rd edition, 1989.

[8] Kiran Lakkaraju, Ratna Bearavolu, and William Yurcik. NVisionIP—a traffic visualization tool for security analysis of large and complex networks. In *International Multiconference on Measurement, Modelling, and Evaluation of Computer-Communications Systems (Performance TOOLS)*, 2003.

[9] Stephen Lau. The spinning cube of potential doom. *Communications of the ACM*, 47(6):25–26, 2004.

[10] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. Portvis: A tool for port-based detection of security events. In *ACM VizSEC 2004 Workshop*, pages 73–81, 2004.

[11] Andreas Noack. An energy model for visual graph clustering. *Lecture Notes in Computer Science*, 2912:425–436, March 2004.

[12] Bryan Parno and Tony Bartoletti. Internet ballistics: Retrieving forensic data from network scans. Poster Presentation, the 13th USENIX Security Symposium, August 2004.

[13] Leonid Portnoy, Eleazar Eskin, and Salvatore J. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, 2001.

[14] S. Staniford, V. Paxson, , and N. Weaver. How to own the internet in your spare time. In *Proceedings of the 2002 Usenix Security Symposium*, 2002.

This work was performed under the auspices of the U. S. Department of Energy by University of California, Lawrence Livermore National Laboratory under contract W-7405-Eng-48.