Intelligent Classification and Visualization of Network Scans

C. Muelder, L. Chen, R. Thomason, K.-L. Ma, and T. Bartoletti

Abstract Network scans are a common first step in a network intrusion attempt. In order to gain information about a potential network intrusion, it is beneficial to analyze these network scans. Statistical methods such as wavelet scalogram analysis have been used along with visualization techniques in previous methods. However, applying these statistical methods causes a substantial amount of data loss. This paper presents a study of using associative memory learning techniques to directly compare network scans in order to create a classification which can be used by itself or in conjunction with existing visualization techniques to better characterize the sources of these scans. This produces an integrated system of visual and intelligent analysis which is applicable to real world data.

1 Introduction

This paper presents an intelligent approach to visual characterization of network scans. This characterization process is a useful tool for analysts in counterintelligence efforts against potential network intruders. Scanning a network is a common first step in a network intrusion attempt. The process of scanning a network is usually performed to determine what exists on a network. For example, if an attacker is looking for exploitable web servers, then he or she would attempt to connect on TCP/UDP port 80 to every possible IP address within a particular range. If there is a web server using port 80 at any of these IP addresses, it will probably respond. However, for addresses where there is nothing, or where there is a computer that is not running a web server, there will be no response. Detecting these scans is fairly

C. Muelder, L. Chen, R. Thomason, and K.-L. Ma
University of California, Davis, e-mail: muelder@cs.ucdavis.edu, leich@ucdavis.edu, rdthomason @ucdavis.edu, ma@cs.ucdavis.edu

T. Bartoletti
Lawrence Livermore National Laboratory, e-mail: azb@llnl.gov

easy (McPherson et al., 2004; Simon et al., 2005), but mining them for information about the attacker can be relatively difficult.

An attacker can do several things in an attempt to make such a scan anonymous; for instance, 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 both malicious and benign scans can often be run on the same port number. For example, both a web crawler looking for new sites and a worm that targets webservers would target port 80. Therefore, some other metric must be used for categorization purposes.

Experimental results have shown that variations in arrival time of the scanning connections often have a high correlation with particular sources (Parno and Bartoletti, 2004). That is, the timing information produces a digital "fingerprint" that correlates to a particular source. While in theory an attacker could try to further obfuscate a scan through controlled delays, in practice this is quite rare. In fact, doing so would require a customized tool which would make the timing signature even more unique. It is surmised that this correlation between timing patterns and sources is due to a combination of factors, including the connection application software employed, the supporting hardware platform, operating system characteristics, and regular interference from other processes on the source system that compete for these resources. Network factors such as number of hops and properties of the routers are certainly responsible for some degree of the timing structure as well, and make this a particularly interesting and challenging problem in network traffic forensics.

The critical question analysts seek to answer is whether the same source ensemble run in an entirely different network (hence, different source IP address), would exhibit a timing structure that is sufficiently similar. This would uniquely identify the environment, and, by extension, the actor behind the observed activity. Alternatively, the analysts would like to know the degree to which the effects of intervening routers produce characteristic packet timing irregularities for different activities conducted from a constant network location. This latter capability would provide a means to determine the veracity of a given source IP address when faced with potential address-spoofing.

To answer this question, analysts must first be able to correlate scans from different source addresses based on timing information. To accomplish this, the analysts need to compare the timing information of very large quantities of network scans quickly and efficiently. Previous methods have used statistical reduction and visualization to compare these scans. However, the statistical reduction has an inherent data loss and can be susceptible to noise, and the direct visualization techniques can become quite unwieldy as the number of scans increases. Intelligent pattern recognition algorithms are quite good at dealing with these issues. They are good at reconstructing distorted or incomplete data, and they scale well to large numbers of inputs.

So, an artificial intelligence based classification methodology was developed that can be used both alone and in conjunction with existing visualization techniques to better characterize potentially hostile scan sources. Since the raw scan data is not directly conducive to comparison, we first transform and encode it into a format that can be compared. We then use a machine learning algorithm to classify the scans. Finally, we combine these results with statistical and visual techniques, and demonstrate how the complement each other.

2 Related Work

The study of network security has been popular for the last decade. Visualization systems have been developed to visualize and compare the network scan pattern in order to detect the potential for attacks. ScanVis (Muelder et al., 2005) presents a means of facilitating the process of characterization by using visual and statistical techniques to analyze the patterns found in the timing of network scans. The system allows large numbers of network scans to be rapidly compared and subsequently identified. Conti and Abdullah (2004) use a parallel coordinates system to display scan details and characterize attacks.

There are many systems designed for detection of scans. There exist visualization based tools such as PortVis (McPherson et al., 2004), as well as data mining methods such as those presented in Simon et al. (2005), but all these approaches focus on the detection of suspicious activity and not on the analysis of such activity. The work presented here does not focus on the detection of these scans. Rather, it focuses on what can be learned once these scans have already been detected.

Machine learning methods – associative memory models in particular – have been widely applied in the pattern recognition and classification area. Tavan et al. (1990) extend the neural concepts of topological feature maps towards self-organization of auto-associative memory and hierarchical pattern classification in 1990. Stafylopatis and Likas (1992) proposed a technique based on the use of a neural network model for performing information retrieval in a pictorial information system. The neural network provides auto-associative memory operation and allows the retrieval or stored symbolic images using erroneous or incomplete information as input.

Machine learning algorithms have also been directly applied to computer security problems in the past. Girardin and Brodbeck (1998) uses a machine learning technique to analyze log files and look for anomalies. These techniques have also been applied to intrusion detection systems in several approaches (Komlodi et al., 2004; Portnoy et al., 2001; Sinclair et al., 1999). These approaches focus on using the machine learning to aid in the detection of malicious activity, rather than the analysis of it. Other approaches use intelligent classification to analyze individual activities to detect patterns in data (Axelsson, 2004a,b). Unlike all these works, the work presented here does not focus on detection. Rather, it aims to analyze potentially malicious activity to gain information about the source.

3 Technical Approach

The methods used in this paper cover a wide variety of techniques, including statistical analysis, visualization, and artificial intelligence, with the ultimate goal of aiding in counterintelligence efforts by characterizing potential adversaries through their scanning activity. An overview of these techniques is shown in Fig. 1. In order to analyze and characterize network scans, they must first be detected and extracted from the raw network data. This is currently being done though the use of some fairly simple statistical data mining techniques. These techniques consist of statistically detecting a scan in progress, then tracking it backwards and forwards in time in order to find when the scan starts and stops. Most of the scans that are detected and extracted in this manner are caused by unknown sources, and so they form a set of unknown scans. But some of the extracted scans originated from known sources, where the parameters of the scans were carefully controlled. That is, properties such as the source hardware, software, scanning tool, and location on the internet are known. This creates a set of controlled scans, which can then be used to compare against the unknown scans.

Due to the constant scanning activity occurring all the time on the internet, the set of unknown scans can rapidly grow to be quite large, so it is infeasible to compare them against each other by hand. The visualization techniques of Muelder et al. (2005) are one set of methods being used to deal with the scalability issues through statistical analysis and visual presentation of both the controlled and unknown scans. These scans can also be used as inputs to an artificial intelligence algorithm, which is the focus of the approach presented here. The controlled scans are used as training data for an associative memory learning process which generates a weight matrix. This weight matrix can then be used in an associative memory reconstruction process to take the unknown scans and classify them. These classified scans can then either be used directly to characterize their sources, or they can be used in the visualization system to improve its effectiveness at scanner characterization.

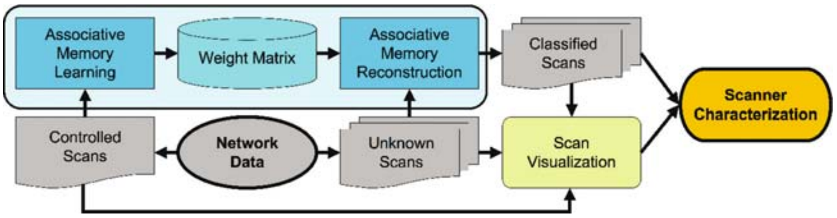


Fig. 1 *The overall methodology.* Known and unknown scans are collected off the network. The known scans are used to train the associative memory in order to create a weight matrix which can then be used to classify the unknown network scans. All of these scans can be visualized, with the overall goal of gaining information about the sources of the scans. The three blue items selected in the upper left are the focus of this paper

3.1 Scan Data and Representation

The scan data used in this paper was generated and collected by the Computer Incident Advisory Capability (CIAC) Group from the network at Lawrence Livermore National Lab (LLNL). The visualization techniques and data formats used are introduced and described in more detail in Muelder et al. (2005). The control data was generated by performing multiple scans, using known tools such as nmap (nmap, 2007) under various settings, on an isolated LAN in order to minimize outside interference. The unknown scans were collected from real network scans targeted at LLNL.

3.1.1 Network Scan Data

Each scan consists of timing information of a scan over a class B network, which contains 65,536 destination addresses. The scan data in its most raw form consists of pairs of destination addresses and times, one for each probe in the scan. Comparing scans in this representation against each other is very ambiguous, since there is no inherent order to this data, so some sort of transformation is necessary. One could choose to order the data according to time or according to address space, and each ordering would yield different results. Alternatively, instead of using the original values, it can often be useful to consider the deltas between subsequent values, in order to emphasize fine details. Various transformations have been performed to create a set of ordered data modes to visualize different aspects of the data (Muelder et al., 2005). There are a plethora of different possible transformations, and several of the modes that were investigated were found to be quite useful in analyzing the scans.

For this research, we chose to use the data mode referred to in our previous work as mode 22 (Muelder et al., 2005). In this data mode each value is derived as the difference between the actual and expected arrival times for the first probe to each destination address, where the expected arrival times are for an ideal scan that starts with the first address, probes successive addresses at a constant rate, and ends with the last address. Because of the fixed size of this transformation, individual scans can be visually represented in detail in a 256×256 grid, where the x and y axes represent the third and fourth bytes of the address, respectively, and the color of each pixel represents the derived value at that coordinate. An example of a scan in presented with this visualization is shown in Fig. 2. As described in the original paper (Muelder et al., 2005), data mode 22 is of fixed size (65,536 values) and is effective at capturing fine details in the scans in the form of intricate patterns. Because of the uniqueness of the patterns under this transformation, data mode 22 is optimal for a pattern matching technique such as is presented here. The techniques described here could be applied to other data modes as well, but that is beyond the scope of this work.

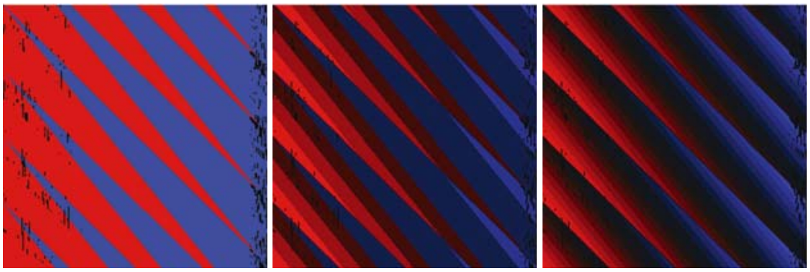


Fig. 2 *Sample network scan using 2, 3, and 4 bits.* The scan is represented by a plot of the third and fourth bytes of the destination IP addresses. Color represents how much earlier or later than expected a probe arrived at that destination. Different bit encodings create different granularities of colors in the scan representation

3.1.2 Binary/Bipolar Coding

The intelligent algorithm we use works on bipolar patterns, which means that every value in the data series is either 1 or -1. Binary patterns can be converted to bipolar patterns easily by replacing every 0 in the pattern with a -1. In order to use non-binary data, such as the real valued mode 22 output, the patterns must first be transformed into binary patterns. The number of bits used can be arbitrary: using higher numbers of bits allows for finer grained distinction between patterns than low numbers of bits, but uses more memory and requires more processing time. Whatever bit length is being used, 0 is reserved for dropped or missed packets, and those values are colored black in the graphs. The highest value, $2^n - 1$ is also reserved and not used. This leaves an equal number of values to represent early packets which are colored red, and late packets which are colored blue. In the simplest coding used in this paper, two bits are used for each IP address in the scan. If the captured time of the first probe is earlier than the expected time, we give the neuron value “01” and if it is later than the expected time we give the value “10”. More bits can also be used to break down the continuous range of possible values into a finer set of discrete ranges. Figure 2 shows a particular scan where the data values are encoded using 2, 3, and 4 bits.

3.2 An Intelligent Method

There are two main categories of machine learning algorithms: unsupervised and supervised. An unsupervised algorithm approaches a data set with little to no preconceived notions about the classes of items that it contains; it attempts to separate the items into statistically distinct groups on its own. Examples of unsupervised algorithms include k-means clustering and self-organizing maps (MacQueen, 1967; Kohonen, 1989). But, unsupervised algorithms cannot effectively use controlled data. Since there is controlled scan data available, a supervised algorithm is more readily applicable to the task presented here. A supervised algorithm is given a large number of examples of items in each class it is to detect; each is labeled with the class to which it belongs. Then, when new items are presented, it puts them into the class they are most likely to belong to. Examples of supervised algorithms include neural networks (McClelland et al., 1986; Werbos, 1974), support vector machines (Cortes and Vapnik, 1995; Cristianini and Shawe-Taylor, 2000), and associative memory (Kohonen, 1978). Of these, associative memory in particular was designed for pattern recognition and has been shown to be faster and more effective then most other approaches for this task (Kohonen, 1978). Since the problem presented here consists of matching large numbers of patterns, associative memory is an ideal choice, and so was selected for our approach.

3.2.1 Associative Memory

The concept of associative memory was first proposed by Kohonen (1978). In the human mind, the memory process takes a stimulus and matches it up with what it remembers about that stimulus. The nature of associative memory is an emulation of human memory, and so it works in a similar way. Associative memory takes a stimulus in an input layer of artificial neurons, feeds them through one or more layers of intermediate artificial neurons, then produces an associated result in an output layer of neurons. The associative memory model has many applications, for example, pattern cognition and reconstruction, image processing, face, character and voice recognition, databases, control systems, and robotics. Human memory is quite robust in that it can correct errors and recognize stimuli even when they are incomplete or distorted. Similarly, in associative memory, given a stimulus pattern that is distorted or incomplete, associative memories are often able to reproduce the correct response pattern. It does this by "remembering" a set of known patterns and then matching up incoming unknown patterns against them. There are a variety of types of associative memory models that have been studied in the last two decades, including Hopfield networks (Hopfield, 1982) and bidirectional associative memory (BAM) (Kosko, 1988).

3.2.2 The BAM Algorithm

Bidirectional associative memory was introduced by Kosko (1988). Like all associative memory algorithms, BAM maps patterns from an input layer X to patterns in an output layer Y, where each layer consists of a set of artificial neurons capable of representing the input or output patterns. BAM has the distinction that it is bidirectional, so it can also map patterns from layer Y back to layer X. The patterns in both layers are represented as artificial neurons which can be either 1 or -1 indicating whether they are firing or not. That is, each element $x_{k,i}$ in each pattern

$\mathbf{x_k}$ in the X layer and each element $y_{k,j}$ in each pattern $\mathbf{y_k}$ in the Y layer is either 1 or -1. While there does exist a continuous version of BAM, it introduces fuzzy logic and non-linearity to the system, so we use the more common discrete BAM here.

BAM is a heteroassociative memory algorithm, which means that the X layer and Y layer of the network have distinct dimensions. This makes it useful for our application since our X layer is very large, while our Y layer can be much smaller. In this work the X layer patterns are the network scans in a bipolar encoding which are of size $m = 65,536 * b$ (where b is the number of neurons used to encode each value) and the Y layer patterns are IDs which are of a size n sufficient to store the number of control patterns we are interested in. While there could be 2^n potential IDs in layer Y, this would not be reliably recalled by BAM (Kosko, 1988). In our system we use 32 neurons in the Y layer by default, but allow the user to set this.

Several methods of generating IDs were tried. The first method was to make each ID a bipolar encoding of the binary representation of an incremental number: $\mathbf{y_1} = (-1,-1,\ldots,-1,-1,1)$, $\mathbf{y_2} = (-1,-1,\ldots,-1,1,-1)$, $\mathbf{y_3} = (-1,-1,\ldots,-1,1,1)$, etc. The next method was to make each ID a bipolar encoding of the binary representation of powers of two: $\mathbf{y_1} = (-1,-1,\ldots,-1,-1,1)$, $\mathbf{y_2} = (-1,-1,\ldots,-1,1,-1)$, $\mathbf{y_3} = (-1,-1,\ldots,1,-1,-1)$, etc. However, the best results were when the IDs were generated pseudorandomly, seeded with the index of the control pattern. The first two methods only use a small portion of the weight matrix because they only use $log(n)$ and n bits, respectively, where n is the number of control scans. The advantage of the third method is that it utilizes the entire range of bits in the Y layer regardless of the number of control scans. This is useful because BAM can get "confused" if scans that are very different have IDs that are very similar (as measured by the Hamming distance). As long as the number of bits used for the ID is large enough, the potential for collisions, or the Hamming distance being too small, can be avoided.

The BAM network itself consists of an $m \times n$ matrix of weights going between each of the m neurons of the X layer and the n neurons of the Y layer. This matrix is derived from the controlled pairs of patterns in the X and Y layers, and then used later to map patterns from the X layer to the Y layer and vice versa. The weight matrix $\mathbf{W}$ is calculated by

$$W_{i,j} = \sum_{k=1}^{K} x_{k,i} * y_{k,j}, \tag{1}$$

where $x_{i,k}$ is the value of the ith neuron in layer X and $y_{k,j}$ is the value of the jth neuron in layer Y of the kth pair of controlled patterns. Calculating this matrix trains the network, which can then be used to classify unknown scans. Since calculating each entry of this matrix is fairly simple, generating the entire matrix can be done fairly quickly.

Classification is performed by starting with a pattern in one layer, then mapping it between the X and Y layers using the matrix until it converges to a constant pair of patterns. Each mapping from the X layer to the Y layer is calculated by

$$y_{k+1,j} = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} W_{i,j} * x_{k,i} > 0 \\ y_{k,j} & \text{if } \sum_{i=1}^{m} W_{i,j} * x_{k,i} = 0 \\ -1 & \text{if } \sum_{i=1}^{m} W_{i,j} * x_{k,i} < 0 \end{cases} \tag{2}$$

for $0 < j \leq n$, where $\mathbf{x_k} = \{x_{k,i} | 0 < i \leq m\}$ is the current pattern in the X layer. And each calculation from the Y layer to the X layer is calculated as

$$x_{k+1,i} = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} W_{i,j} * y_{k,j} > 0 \\ x_{k,i} & \text{if } \sum_{j=1}^{n} W_{i,j} * y_{k,j} = 0 \\ -1 & \text{if } \sum_{j=1}^{n} W_{i,j} * y_{k,j} < 0 \end{cases} \tag{3}$$

for $0 < i \leq m$, where $\mathbf{y_k} = \{y_{k,j} | 0 < j \leq n\}$ is the current pattern in the Y layer. Eventually this system will converge (as shown by Kosko (1988)), and the resulting patterns are output. In our case, the input is one of the scans into the X layer, and once it converges, the output is the resulting ID in the Y layer. If the output exactly matches the ID of one of the control patterns a match has been successfully made, otherwise it remains unclassified.

3.2.3 BAM Results

Application of the BAM algorithm to actual network scans yields valuable results. As depicted in Fig. 3, we found that the BAM algorithm, when trained on a single example of multiple kinds of patterns, effectively categorizes the unknown scans. Even when patterns are somewhat different due to varying amounts of noise or distortion, they are still associated with the correct training pattern. This shows that the techniques presented here are able to effectively classify many network scans according to a set of training scans. Thus, given a good set of training scans, this technique can classify large numbers of unknown scans.

But classification in this manner is limited by the control data that is available. For example, this technique could not be used to detect and classify a new worm, since no control data would be available for that particular worm. Even creating control data for known malicious attacks such as worms can be difficult due to policy issues regarding launching attacks against one's own network. However, the techniques presented here would easily extend to any kind of scan-like data provided one has access to or can generate such control data. Additionally, if there are a few particular unknown scans that warrant further investigation, the BAM algorithm could

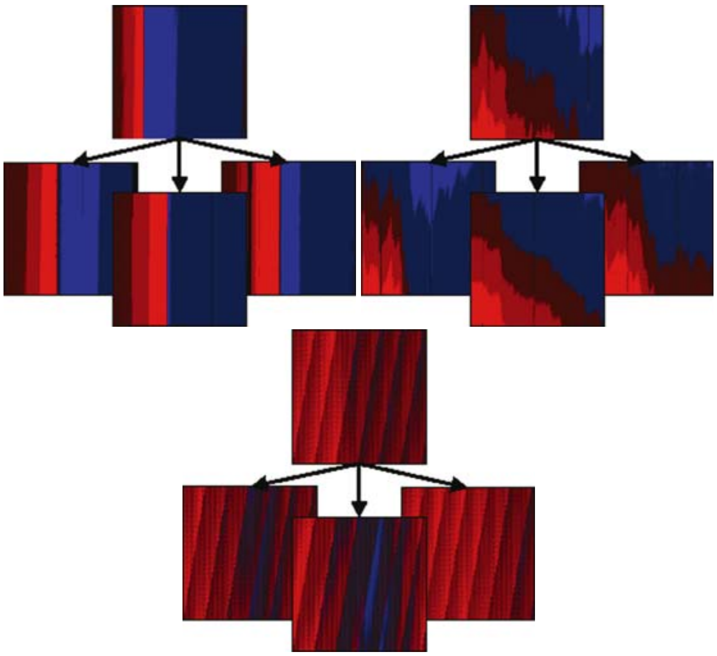


Fig. 3 *Classification examples.* The system matches numerous scans to their corresponding training scans. Three training patterns using 3 bits for encoding are shown. Below each are several example classifications that were made

be trained on those select unknown scans to try and determine if similar scanning behaviors have happened in the past.

3.3 Visualization Integration

Once scan patterns have been classified by matching them up with their associated control patterns, it is useful to present these results visually to the user. This is particularly important as the number of classified scans increases and becomes too unwieldy to analyze by hand. For this work, it was decided that these results could best be visualized as a coloring of an existing graph generated by statistical means as in ScanVis (Muelder et al., 2005). So, the scans were run through a wavelet analysis to generate scalograms, and these scalograms were compared against each other to generate a graph representation.

The combination of the intelligent and statistical techniques is effective because the wavelet techniques and the associative memory techniques measure different properties of the data. For instance, the wavelet analysis can detect a particular feature within a pattern, but would not carry any information about where in the scan this feature occurs. The associative memory, on the other hand, would reliably match scans based on where a feature occurs in a scan, but would also match a slightly different feature that occurs in the same area. Thus, the two approaches can complement each other well when used together.

3.3.1 Wavelet Analysis

While transforming the scans into a regular form makes them comparable, a direct comparison would still not yield useful results. For instance, if an attacker scanned every other network address one day, then came back the next day and scanned the addresses that were skipped the first time, then a direct comparison would reveal no similarity, even though the patterns would be nearly identical; they would just be out of phase from each other. However, there are several algorithms utilizing frequency analysis that are useful for handling this kind of data, such as Fourier transforms and wavelet analysis. 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. These are things that Fourier analysis has been found to handle poorly (Graps, 1995).

So, wavelets are used because they are relatively resistant to both phase shifts and noise. This means that similar patterns will have similar wavelet scalograms, even if the patterns are shifted slightly or different parts of the pattern are missing, as can be seen in Fig. 4a. But dissimilar patterns will still produce different scalograms, as can be seen in Fig. 4b. There are several variations on the wavelets that can be used, several of which are enumerated in (Muelder et al., 2005). For this work we chose to use the following wavelet. Given a series of $N = 2^n$ items $D_0 = (d_{0,1}, d_{0,2}, ..., d_{0,N})$, we calculate recursively:

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

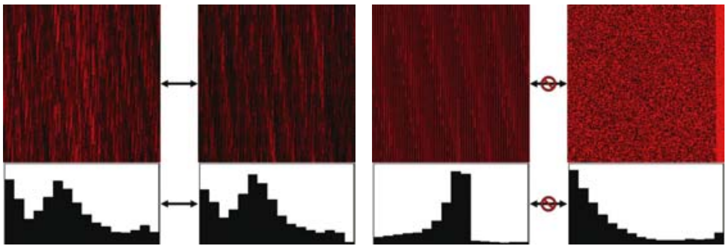

(a) Similar patterns (b) Dissimilar patterns

Fig. 4 *Wavelet scalograms.* Applying wavelet scalograms creates a representation that is directly comparable, at the cost of data loss (Muelder et al., 2005)

for $0 < k < n$, where

- $d_{k,i} = \frac{d_{k-1,i}+d_{k-1,i+1}}{2}$,
- $s_{k,i} = \begin{cases} -1 & \text{if } d_{k-1,i} > d_{k-1,i+1} \\ 1 & \text{if } d_{k-1,i} < d_{k-1,i+1} \\ 0 & \text{if } d_{k-1,i} = d_{k-1,i+1} \end{cases}$.

At each recursion the σ values are the mean of the corresponding data series, and they estimate the variance at each resolution. Once these wavelet scalograms are calculated, they can be directly compared to one another. The downside to using these wavelets is that they lose fine details in the data, such as where a particular feature occurs in the pattern. Still, they can be used to create a good approximate overview from which other information can be shown.

3.3.2 Graph Representation

In order to represent large numbers of these scans at once, a graph representation of the scans was used, just as in Muelder et al. (2005). In this view, each scan is a node in a complete graph, and each edge is calculated as the inverse Euclidean distance squared between the wavelet scalograms of the two scans it connects. That is, for an edge between scans **A** and **B**, the weight $W(\mathbf{A},\mathbf{B})$ is:

$$W(\mathbf{A},\mathbf{B}) = \frac{1}{1+||(\mathbf{A}-\mathbf{B})||^2}.$$

A force-directed layout, LinLog (Noack, 2004), is then applied to this graph, which groups similar scans together according to the edge weights between them. Edges that are below a threshold are dropped for clarity. This creates an overview in which large scale trends in the wavelet analysis, such as clusters, can be seen.

We can also use it to display the results of applying the BAM algorithm by coloring the nodes according to their classification. An example of this is shown in Fig. 5, where the classification process was run on a small set of controlled scans. In this example, the set of scans was classified according to three training scans, of which several samples are shown. In this image it can be seen that scans that have the same color usually have very similar patterns. It can also be seen that some scans did not match any of the controlled patterns, which indicate anomalous patterns which could be looked at more closely.

However, of even more interest is that the BAM classification and the clusters according to wavelet analysis do not have a 1-to-1 correlation. There are more than one cluster that are colored the same according to BAM, indicating a difference between these scans that wavelet analysis picked up that BAM did not. And there are clusters that contain a scan or two of a different color than the rest of the scan, indicating scans that contain a difference that BAM detected that wavelet analysis missed. Thus, the two techniques complement each other well in this manner.

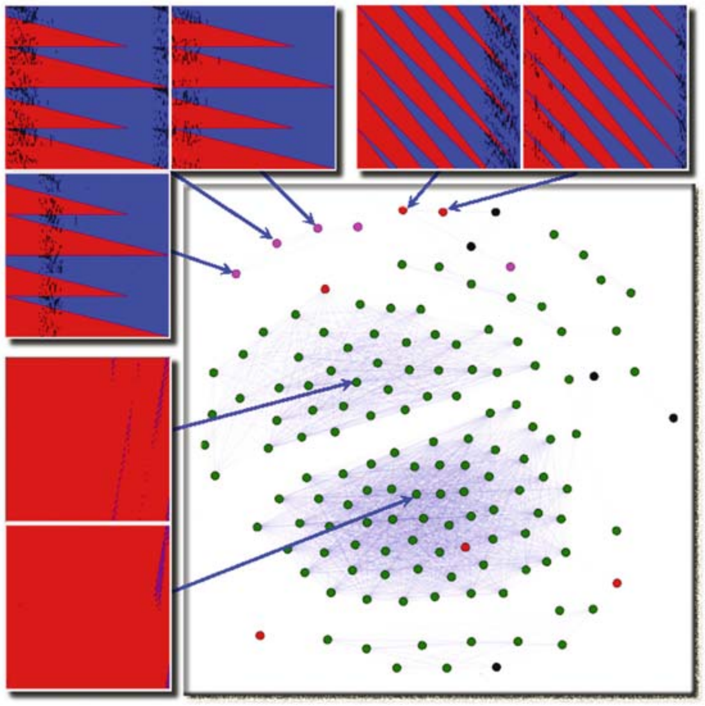


Fig. 5 *A graph representation.* A graph of a set of controlled scans is generated and laid out according to the wavelet scalogram analysis, as in ScanVis (Muelder et al., 2005). BAM was used to classify these control scans according to three selected scans. The results of using BAM is represented by coloring the nodes of this graph. Examples of scans from each classification are shown around the outside. Outliers which were not classified by BAM are shown as black nodes

3.4 A Case Study

Figure 6 shows the results of applying the techniques presented here to color a graph of over 800 unknown scans collected off the network. One pattern that is readily apparent is the large number of unclassified scans, which are colored black in the image. This is indicative of the large number of different kinds of scans active on the internet that have not yet been identified. However, by applying an associative memory approach, these scans have been highlighted by a process of elimination, indicating a set of scans that would be beneficial to look at in more detail. Using wavelets alone would not have identified these scans.

Another prevalent pattern is that the nodes that were clustered by the statistical analysis are generally classified the same according to the associative memory. What is of interest is that some of the scans within a cluster are classified differently. This is indicating that the machine learning algorithm is picking up on a detailed pattern that is lost by this statistical analysis. As is shown in Fig. 7, the difference in the patterns is not necessarily even difficult to discern by eye. This shows that machine

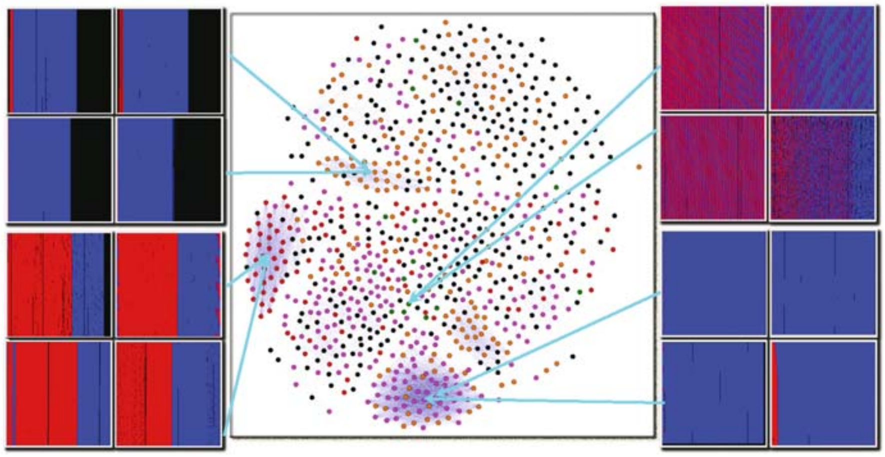
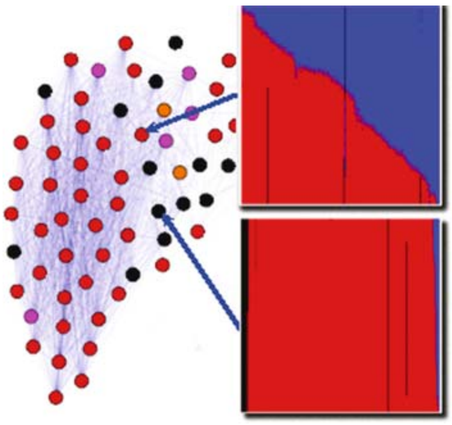


Fig. 6 *A graph showing the results of the BAM classification to unknown scans.* As in Fig. 5, a graph was made as in ScanVis (Muelder et al., 2005) and the nodes are colored according to a BAM classification. In this case, BAM was trained with four different patterns, so it classified nodes into four groups. Examples from each classification are shown. Outliers are shown as black nodes

Fig. 7 *A close up of one cluster.* Most nodes are colored red, and their detail view looks like the detail view on top. But some are not, and look different as is shown in the detail view at bottom. This distinction was not picked up by the wavelet analysis



learning techniques can quickly lead the analyst to find pattern details that were lost by the statistical analysis.

4 Future Work

One possible extension of this work would be to investigate looking at the differences between the original scans and the controlled scans that the associative memory classified them as. That is, by subtracting out the part of the pattern that is controlled, it should be possible to isolate the distortion patterns that are created by uncontrollable effects such as router delays. Doing so should greatly enhance the capability to correlate and identify such common but minute effects, even when the underlying scans are vastly different due to large scale effects such as the scanning tool used.

Another aspect that could be considered is the application of unsupervised algorithms to this task. These would produce a clustering without the need for controlled data. This could lead to an even more effective tool for finding outliers in the graph, since each true outlier would have a unique color instead of all being grouped into one class.

Finally, it would be useful to pursue a tighter integration of intelligent techniques such as BAM with visualization systems such as ScanVis. We have demonstrated the effectiveness of such a combination in one direction, by presenting the results of applying the BAM algorithm to a graph such as that in ScanVis. It could be useful to go the other direction, and take some information learned from ScanVis and quickly feed it into the training of an algorithm such as BAM. That is, when an anomaly or trend is detected, it would be convenient to be able to select the scans of interest as training data for BAM directly from within ScanVis, and have BAM subsequently classify the rest of the graph automatically.

5 Conclusions

In the system presented in this paper, a machine learning method, BAM, has been used to perform intelligent network scan pattern reconstruction and classification. When given a set of controlled scan patterns and a noisy or incomplete pattern, the results show that the system can successfully return the complete pattern. These restored patterns are much more convenient for further studies, such as pattern comparison or pattern clustering, for the purpose of correlating malicious network activities. This paper has also shown the effectiveness of combining machine learning techniques with existing visualization and statistical techniques to create a useful visual representation for dealing with large numbers of network scans. Therefore, the results naturally lead to the feasibility of applying associative memory models in reconstruction and recognition of network scan patterns.

From an operational standpoint, one of the most important tasks in cyber security is "damage assessment". When damage (a successful intrusion, or a detected data-exfiltration activity) is discovered at one point, security managers must immediately seek evidence of this activity more broadly, where it may not yet have been discovered. Unfortunately, this assessment is typically limited to a search for the same (outsider) IP address, under the naive assumption that the hostile agent will be using the same IP address for their activities directed at range of targets. If the intruder or outside agent employs different IP address, either in time (evolving) or in space (intentionally to avoid naive correlation) then these broader damage assessments will fail.

A system by which an intruder's "pattern", such as exemplified by scan timing characterization, could be used to quickly "re-identify" an intruder (irrespective of IP address) would greatly enhance the effectiveness of damage assessment activities, enabling detection where none was previously possible. Our system demonstrates that associative memory techniques and visualization together form an effective basis for such a characterization system. The system could quickly be trained upon the known intruder evidence, and then sought more broadly to determine the true scope of damage.

Acknowledgements This research is sponsored by the NSF CyberTrust program.

References

Axelsson, S.: Combining a bayesian classifier with visualisation: understanding the ids. In: VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 99–108 (2004a)

Axelsson, S.: Visualising intrusions: Watching the webserver. In: Proceedings of the 19th IFIP International Information Security Conference (SEC2004). IFIP, Tolouse, France (2004b)

Conti, G., Abdullah, K.: Passive visual fingerprinting of network attack tools. VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 45–54 (2004)

Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). DOI http://dx.doi.org/10.1023/A:1022627411411

Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge, UK (2000)

Girardin, L., Brodbeck., D.: A visual approach for monitoring logs. In: Proceedings of the 12th Usenix System Administration Conference, pp. 299–308 (1998)

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

Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. In: Proceedings of the National Academy of Sciences, vol. 79. National Academy Press, Washington, DC (1982)

Kohonen, T.: Associative Memories: A System Theoretic Approach. Springer-Verlag, Berlin Heidelberg New York (1978)

Kohonen, T.: Self-Organization and Associative Memory. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1989)

Komlodi, A., Goodall, J., Lutters, W.: An information visualization framework for intrusion detection. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI) (2004)

Kosko, B.: Bidirectional associative memories. IEEE Trans. Syst. Man Cybern. **18**(1), 49–60 (1988)

MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Symposium on Math, Statistics, and Probability, vol. 1, pp. 281–297 (1967)

McClelland, J.L., Rumelhart, D.E.: The PDP Research Group: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA (1986)

McPherson, J., Ma, K.L., Krystosk, P., Bartoletti, T., Christensen, M.: Portvis: A tool for port-based detection of security events. In: ACM VizSEC 2004 Workshop, pp. 73–81 (2004)

Muelder, C., Ma, K.L., Bartoletti, T.: A visualization methodology for characterization of network scans. Visualization for Computer Security, IEEE Workshops, pp. 4–4 (2005) nmap: (2007). URL http://insecure.org/nmap/

Noack, A.: An energy model for visual graph clustering. Lect. Notes Comput. Sci. **2912**, 425–436 (2004)

Parno, B., Bartoletti, T.: Internet ballistics: Retrieving forensic data from network scans. Poster Presentation, the 13th USENIX Security Symposium (2004)

Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: ACM Workshop on Data Mining Applied to Security (DMSA) (2001)

Simon, G., Xiong, H., Eilertson, E., Kumar, V.: Scan detection: A data mining approach. Technical Report AHPCRC 038, University of Minnesota – Twin Cities (2005)

Sinclair, C., Pierce, L., Matzner, S.: An application of machine learning to network intrusion detection. In: ACSAC ’99: Proceedings of the 15th Annual Computer Security Applications Conference, p. 371. IEEE Computer Society, Washington, USA (1999)

Stafylopatis, A., Likas, A.: A pictorial information retrieval using the random neural network. IEEE Trans. Software Eng. **18**(7), 590–600 (1992)

Tavan, P., Grubmuller, H., Kuhnel, H.: Self-organization of associative memory and pattern classification: recurrent signal processing on topological feature maps. Biol. Cybernet. **64**(2), 95–105 (1990)

Werbos, P.J.: Beyond regression: New tools for regression and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University, Division of Engineering and Applied Physics (1974)