Multiple Self-Organizing Maps for Intrusion Detection

Abstract

The Kohonen self-organizing map is an extremely powerful mechanism for automatic mathematical characterization of acceptable system activity. Because it spontaneously develops a sophisticated characterization of the system whose behaviors it is trained to recognize, it could detect intrusions which it has never observed simply by noting the degree to which they differ from normal activity. After discussing the design of a network monitoring system which would maximize the potential of the self-organizing map, we describe briefly our experimental results in which a simpler system resoundingly detected two different exploits which we perpetrated against one of our servers.

Keywords: Intrusion detection, misuse detection, neural networks, self-organizing maps.

Brandon Craig Rhodes, James A. Mahaffey, James D. Cannady

Computer Science and Information Technology Division
Information Technology and Telecommunications Laboratory
Georgia Institute of Technology
Atlanta, Georgia 30332

Brandon Craig Rhodes 404 894-0859  brandon.rhodes@gtri.gatech.edu
James A. Mahaffey 404 894-8967  jim.mahaffey@gtri.gatech.edu (point of contact)
James D. Cannady 404 894-9730  james.cannady@gtri.gatech.edu
Multiple Self-Organizing Maps for Intrusion Detection

Abstract

The Kohonen self-organizing map is an extremely powerful mechanism for automatic mathematical characterization of acceptable system activity. Because it spontaneously develops a sophisticated characterization of the system whose behaviors it is trained to recognize, it could detect intrusions which it has never observed simply by noting the degree to which they differ from normal activity. After discussing the design of a network monitoring system which would maximize the potential of the self-organizing map, we describe briefly our experimental results in which a simpler system resoundingly detected two different exploits which we perpetrated against one of our servers.

Keywords: Intrusion detection, misuse detection, neural networks, self-organizing maps.

1 Introduction

Intrusion detection systems have been an active area of research and development since 1987[Den87]. As computer networks have become faster and more complex, the need for an effective intrusion detection approach has increased. This paper describes an innovative approach to intrusion detection that uses a self-organizing neural network to recognize anomalies in a computer network data stream. Unlike most existing approaches, the research described in this paper focuses on the design of a method of recognizing potentially intrusive activity in network traffic without the need for pre-defined attack signatures that limit the effectiveness of expert system-based approaches. The following sections provide an overview of existing intrusion detection approaches, a discussion of neural networks and the application of the technology to intrusion detection, and a description of the ongoing research effort that is applying multi-level self-organizing neural networks in the detection of network attacks. 1

2 Prior Neural Network Approaches

Effective intrusion detection is a difficult and elusive goal for system administrators and information security researchers. The inherent complexity of computer systems, the variety of potential vulnerabilities, and the skill of many attackers combine to create a problem domain that is extremely difficult to address.

Intrusion detection has traditionally been focused on one of two approaches. Anomaly detection identifies activities that vary from established patterns for users, or groups of users. It typically involves the creation of knowledge bases that contain the profiles of the monitored activities. The second approach, misuse detection, involves the comparison of a user’s activities with the known behaviors of attackers attempting to penetrate a system. While anomaly detection typically uses threshold monitoring to identify anomalies, misuse detection is most often accomplished using a rule-based approach.

1 This work is supported by Army Research Laboratory DAKF11-97-D-0001-0036.
Expert systems are the most common form of rule-based intrusion detection approaches. Unfortunately, expert systems have little or no flexibility. Even minor variations in an attack sequence can affect the activity-rule comparison to a degree that the intrusion is not detected by the intrusion detection mechanism. Some approaches have increased the level of abstraction of the rule-base to provide a partial solution to this weakness, but this simultaneously reduces the granularity of the intrusion detection process.

2.1 Neural Networks

An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to adapt to the desired outputs. Unlike expert systems, which can provide the user with a definitive answer if the characteristics that are reviewed exactly match those which have been coded in the rulebase, a neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics which it has been trained to recognize.

2.2 Neural Network Intrusion Detection Systems

An increasing amount of research has been conducted in the last few years in applying neural networks to intrusion detection. If properly designed and implemented neural networks offer the potential to address a number of the problems encountered by rule-based approaches to intrusion detection. Neural networks were specifically proposed to identify the typical characteristics of system users and identify statistically significant variations from the user’s established behavior. Fox[FHR90] used a SOM to learn the characteristics of normal system activity and identify statistical variations from the norm that may be an indication of a virus.

The first use of a SOM in misuse detection was described in [CM98]. The prototype neural network approach was designed to address the need to identify temporally dispersed and possibly collaborative attacks in a simulated data stream. Temporally dispersed attacks are those conducted by a single attacker over an extended period of time and collaborative attacks are conducted by multiple attackers working in concert to achieve a single intrusion. Each of the attackers actions taken individually may appear innocuous with the attack becoming apparent only if all of the events are viewed together. Both of these types of attack may be very complex. As a result an enhanced pattern recognition approach was designed and tested to identify series of events that constitute an attack. The tests conducted on the hybrid neural network indicated that the prototype was very effective in identifying the simulated attacks.

3 Kohonen’s Self-Organizing Map

To characterize communications traffic on a network, our research employs a specialized neural network called a self-organizing map. These networks automatically categorize and arrange input items by similarity, creating a mathematical landscape whose terrain characterizes the entire range of item types.

The prototypical neural network, the sort that springs to the imagination when the term is mentioned, consists of a large number of fairly simple processing elements con-
connected by weighted interconnections — often forming distinct layers cascaded in series. Such networks take as input a vector of numeric values which, when presented at the input element layer, act through the weighted connections to determine the values taken by elements of each subsequent layer. The values assumed by the last layer are taken as the network’s output. Thus the network is the implementation of some function, mapping each input vector to a corresponding output value or vector.

Most neural network architectures are based on supervised learning. During an initial, dedicated training period, the network is repeatedly presented with series of input-output associations that its instructor wishes it to learn. An algorithm iteratively adjusts the network weights to produce results more closely approximating these associations. Training concludes when the network displays satisfactory accuracy at producing the desired output when presented with each input.

The self-organizing map was developed by Kohonen [Koh95] may appear at first glance to be degenerate. Its nodes are organized in a single two-dimensional layer lacking interconnections of any sort — the relationships between adjacent nodes must be established through lateral feedback applied during training. Each node in the map is a vector of the same size as the input vector, and the output of the map is computed by simply comparing the input vector with every node and outputting the location of the most similar node.

Training begins on a map whose nodes have been assigned random values across the range anticipated for the corresponding elements of the input vector. The map is queried for the best-fit location of each vector in a training set, and the nodes in the neighborhood of each winner are adjusted to even more closely fit that training vector. This process is repeated with successively smaller definitions of the area defining the ‘neighborhood’ of a node, until the positions assigned to the training vectors have become stable and show good fit to the nodes of the map.

The result of this process, which was extensively analyzed by Kohonen, is that the training vectors are organized according to mutual similarity. Groups of very similar vectors cluster together on the map, while subtle variations often result in vectors arranged in finely graded lines or arcs. The distance between any two training vectors represents not their absolute dissimilarity — which we already knew how to compute anyway — but their difference when considered in the space of possibilities enumerated by the training set.

When a trained Kohonen map assigns a location to a novel input vector, it would be possible to interpret the output by assigning meaning to the various clusters and regions staked out by the training vectors. While this process could be roughly automated, perhaps by using known vectors as tracers to help identify the clusters, it would be difficult to draw intelligent distinctions in regions of the map where clusters graded together or where vectors were sparse. This application might be more appropriate for information visualization than for automated processing such as we consider in this paper.

Instead of considering where on the map an input vector is placed, we can consider how well it fits on the map at all. Since each vector is placed near the node to which it bears the greatest similarity, the measure of that similarity tells us how characteristic the vector is of the values given in the training set. If the training vectors represent
the entire range of normal behavior for a system, then deviations from normality will manifest themselves as vectors which fit the map very poorly.

4 Applying Kohonen’s Map

The map interpretation scheme just presented has appeared in previous work on security monitoring. Earlier work employed it to characterize normal machine state, and thus make it possible to detect misuse affecting system processes and resources[FHR90]. We instead use it to identify abnormal network traffic that might indicate attempts to abuse or compromise the services offered by networked hosts.

More recent work[CM98] has employed the Kohonen map in detection of misuse events that develop over time. There the map was used to categorize incoming events for presentation to a conventional feed-forward neural network, where the categorization was performed by attempting to diagnose which cluster each incoming packet belonged to when plotted on the two dimensional surface of the map.

While earlier work considered Kohonen maps that processed the entire state of a system or network, our work breaks ground by using collections of more specialized maps. By introducing deterministic preprocessing of network traffic, we narrow the load placed on each map and, we argue, facilitate the construction of a system that is much more sensitive to abnormalities in the content of network traffic.

4.1 Deploying Specialist Maps along a Monitor Stack

Inspection of the network link serving a large facility will typically find it packed with heterogeneous traffic. There are several sources of this variety:

- Packets are not only destined for different hosts, but many belong to particular conversations ongoing between applications on those hosts. Other packets may not belong to conversations, instead carrying asynchronous unidirectional communiques that receive no acknowledgement.
- Each packet bears a series of headers that each direct the operation of one of the protocols which the communicating agent is employing to transmit his information.
- The pace and duration of connections is quite variable. Automated transactions such as nameserver and HTTP requests may often take less than a second, while login sessions operated directly by humans may persist for hours.
- The payloads carried by the observed packets will vary remarkably between applications. Some will hold exactly formatted text, others encode the painstaking line editing of users at their keyboards, and still others will communicate arbitrary binary data.

It would be unreasonable to expect a single Kohonen map to usefully characterize such disparate information.

These considerations motivate the construction of a monitor stack, utilizing protocol analyzers to reduce and segregate the traffic before it is subjected to map analysis. Since packets are produced by layered protocols in the originating host, the monitors are similarly layered — allowing the extraction of statistics about lower protocol layers while permitting the reconstruction of the data streams they support.
Note that the completeness of the monitoring stack output is important. Malicious activity can be targeted at any protocol layer. An otherwise innocent web browser request could be degenerately fragmented at the IP layer in an attempt to crash the receiver’s operating system; floods of apparently normal connection requests could monopolize the resources of a server’s Transmission Control Protocol (TCP) layer to deny service to legitimate users; or, an abnormally long and garbled file name requested from a file server could induce it to run arbitrary code on behalf of an intruder.

Designing an analyzer stack for reconstructing network activity is not novel. But it is an essential vehicle for our research because, by illustrating the kinds of information available from a complete decomposition of network traffic, it provides a context in which we may consider the intelligent deployment of neural networks.

It is strongly suggested by this architecture that each neural network be a kind of specialist, trained to recognize the normal activity of a single protocol and ready to raise an alarm when a significant deviation is detected. The specialization could be carried even father, with the activities of particular remote hosts or individual users being subjected to the analysis of neural networks honed to recognize their typical behavior.

Of course many desirable monitors can, and should, be constructed without neural networks — many such schemes are already operating in extant intrusion detection systems. It would be ridiculous to attempt a neural approach to recognizing, say, an attack based on the reassembly of malformed IP fragments, when simple conditional checks on fragment offsets and lengths will accomplish this flawlessly and very efficiently. Rather, we are seeking out those roles in the network protocol hierarchy where flexible pattern recognition is a significant asset over more traditional approaches.

Note that protocol decomposition can continue for as long as we can invoke further deterministic rules to guide it. Not only could the data stream reconstructed from TCP packets be further interpreted as a Simple Mail Transport Protocol transaction, but the mail message itself could be processed for MIME encoded attachments that might include documents with macros. The computing resources available for monitoring a given network link will of course place limits on how much processing can be performed, but we expect that many facilities would find it worthwhile to configure certain monitors to drill down quite far into the structure of transactions that have posed earlier security problems.

4.2 Vectorization Options

Our discussion to this point has glossed over the issue of how incoming data streams are packaged for presentation to a Kohonen map, which (as was noted in the earlier discussion of them) accepts fixed-length vectors as input.

The invention of schemes by which packets, transactions, and data streams might be represented numerically represents both the greatest challenge and the greatest opportunity of this research. The choice of which features to represent, and how to translate them into numbers, will unavoidably involve highlighting certain aspects of network activity while denigrating others. The task for the designer of each detector will be to develop vectorizations which increase, as much as is possible, the contrast between innocent and malicious activity.

Obviously the development of multiple successful vectorization techniques for a
given protocol suggests the deployment of multiple Kohonen maps, each viewing the network activity through the lens of a different vectorization. While it may often be possible to construct an intrusion that appears normal to one particular map, the presence of several maps — each watching a different aspect of the protocol’s behavior — would make it extremely unlikely that an intrusion could pass unnoticed.

It is not clear at this stage in our research how to decide if features should be combined into single vectors, or separated and presented in separate vectors to different maps. If too many features are combined into a single vector, we run the risk that an intrusion will not affect enough of them to register as anomalous; but if every feature is presented to its own map, then we lose the ability to detect unusual correlations. We hope in the course of our research to develop guidelines for detector construction that address these questions.

A final degree of freedom in our design is provided by the meaning of the “similarity” between two vectors, which you may have noted we left undefined. This is because, while the computation of Pythagorean distance is the most familiar measure of similarity between two vectors, it is neither the only such measure nor even guaranteed to be the best for the application we are considering. Other techniques available include: taking a weighted sum of the difference between corresponding vector components; computing the editing distance, or some other measure of textual similarity, between strings encoded in the vectors; and statistical techniques such as taking ratios between various quantities or deriving means or averages.

Thus a measure of similitude must be chosen or developed for each vectorization method, since it is this measure that the Kohonen map will use to interpret the meaning of the vector elements.

5 Experimental Evidence

We have constructed an anomaly detector based on a Kohonen map, and successfully detected two different exploit attempts we perpetrated against our own server. The following is an outline of our experiment.

5.1 Design

We have implemented a Kohonen map using the C language on a Linux workstation, and a test harness written in Python that includes basic visualization facilities allowing us to monitor the training and use of the map. All of our network data was collected locally using the `tcpdump` packet capture program that is standard equipment with many versions of Unix.

We chose to monitor requests to our Domain Name Service (DNS) port because of the relative simplicity of its protocol: its requests and replies are defined to be symmetric, so a single routine can be used to parse both. DNS implements a worldwide distributed database for performing host name resolution, allowing users to use names like “mit.edu” rather than raw IP addresses like 18.72.0.100. We restricted our attention to requests using the TCP protocol, since this is the protocol used by both of the exploits which we possess.2

---

2We employed both the Riders of the Short Bus (rotshb) and bind4-9-5 exploits, which can be downloaded from the Packet Storm security archive (visit http://packetstorm.securify.com/exploits/apps/bind/).
From a sample of roughly forty packets we used the first thirty to train the map, and the other dozen to determine how well the map generalized from these thirty examples. A listing of all forty vectors, sorted by how well they fit on the map, showed that the map did not noticeably favor the thirty vectors from the training set. We then submitted the packets from both exploits to the trained maps and measured how well they fit.

Since both of the test intrusions were buffer overflow attempts, we designed vectors consisting of a simple six-category histogram indicating how many octets each packet fit a particular character class (such as alphabetic, numeric, control, and non-ASCII). Since buffer overflow arguments tend to be unusually large and place binary data where text is expected, we hypothesized that this vectorization scheme would make buffer overflow attempts particularly striking.

Note that we have not at this point attempted to reconstruct the TCP packets into a data stream, as the system envisioned above would do. Since we instead simply consider each packet individually, our results may be more modest than would be achieved by a more complete monitor stack.

5.2 Results

We were extremely encouraged by the stability of our self-organizing map. Since each map is initialized randomly, and reaches a unique spatial arrangement for the training vectors, we did not know whether our measure of how well each vector fit would also vary substantially. Our training runs have to date produced quite consistent measures, rarely approaching a ten percent difference between the measures of fit for the same vector on different runs. So while the location of the clusters is different each run, the properties of the topological habitat they carve out are eminently reproducible.

In the case of our data and our particular distance measure, all of the “normal” traffic scored somewhere between zero and three, save for one outlier that consistently registered around thirteen — not a very good fit.

The bind4-9-5 exploit produced encouraging results. Of the roughly seven packets transmitted to accomplish the exploit, two registered just above eighty, indicating they did not fit well on the map at all, and two other registered around six hundred thirty — indicating an extreme anomaly. This ratio of more than fifty between the worst-fitting training vector and the vectors produced from intrusive packets would provide ample leeway for establishing an alarm threshold in production system.

The results from the rotshb exploit were even more extravagant; not only did it also display two packets whose fitness measure landed at just above eighty, but the two packets actually carrying the exploit payload scored a whopping thirteen hundred and sixty each!

These experiments have provided strong encouragement for the implementation of a working intrusion detection system utilizing self-organizing maps.

6 Conclusion

The Kohonen self-organizing map is an extremely powerful mechanism for automatic mathematical characterization of acceptable system activity. We have argued that it should be applied to the analysis of data collected from network monitoring, and have
designed a monitoring system that would preprocess network packets to highlight their properties for inspection by a self-organizing map.

Our actual experiments show that even a simple map, when trained on normal data, will detect the anomalous features of both buffer overflow intrusions to which we exposed it. The ratio by which normal and intrusive packets differed in the self-organizing map was computed, and found to be greater than an order of magnitude in both instances.

This approach is particularly powerful because the self-organizing map never needs to be told what intrusive behavior looks like. By learning to characterize normal behavior, it implicitly prepares itself to detect any aberrant network activity.

References


