Combining Self-Organizing Map Algorithms for Robust and Scalable Intrusion Detection

Zuerst erschienen in:
Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06) - Volume 02
Combining Self-Organizing Map Algorithms for Robust and Scalable Intrusion Detection

Sahin Albayrak  
DAI-Labor  
(TU Berlin) 
sahin@dai-lab.de

Achim Müller  
Deutsche Telekom - Laboratories 
achim.mueller01@telekom.de

Christian Scheel  
DAI-Labor  
(TU Berlin) 
scheel@dai-lab.de

Dragan Milosevic  
DAI-Labor  
(TU Berlin) 
ragan@dai-lab.de

Abstract

In the field of intrusion detection systems, the aspect of anomaly detection is very important, and consequently there are many approaches that address these security issues. The usage of Self-Organizing Map (SOM) makes a foundation for some of these approaches, which consequently often have problems to cope with the requirements of huge nowadays networks. The proposed approach focuses on improving the usage of SOMs for anomaly detection, by combining the strengths of different SOM algorithms. The performed evaluations have shown the necessity of paying attention to different aspects, coming along with network nodes, to individually choose the best matching SOM for each node’s anomaly detection.

1. Introduction

Anomaly detection is an important part of intrusion detection and protection. Intrusions are unauthorized actions on a system done by users or programs. Usually intrusions done by programs like Trojan horses have different patterns and characteristics which can help to detect them. These characteristics can be measured in network nodes, hard drives, processors and every other component which may be affected by an intrusion. As a result the affected component will be in an anomaly state, being by definition a deviation from the normal, common order or form or rule.

But what if the characteristics of such programs are not known by the system? Then there is a need for detecting attacks without knowing individual conditions of that aggression. The characteristics are still present, but unknown. So the system has to classify one such state as abnormal by itself. Unfortunately, a system is often aware of only normal states, whereas attacks are exceptions, and usually there are no depending measured data about them. That leads to the natural idea of learning the normal states to detect the anomalous ones. Each measuring which sheers away from the normal and behaves suspiciously will be reported to higher instances for further inspection. The problem is to learn how to make distinctions between normal and abnormal.

In the area of intrusion detection systems, the use of unsupervised learning algorithms is going to support the detection of anomalies [18] [23]. For two main reasons the detection should be done by the Self Organizing Map (SOM) algorithm, created by Kohonen [17]. First this neural network approach is unsupervised and does not need any feedback from the user, and second these Maps can be visualized to clarify why a state was declared as anomaly.

To achieve even better protection of computer networks, various SOM algorithms can be combined, as it is going to be shown in this paper. In the following sections, after critically presenting drawbacks of other attempts to provide anomaly detection, one scenario will be used to illustrate main security challenges. The core of this paper is then contained in section 5, which gives principles being in the basis of the used protection approach that creates the mapping of different SOM algorithms to every network node. This paper concludes with sections where evaluation details and ideas for future work are presented.

2. Related Work

The importance of an adequate intrusion detection has resulted in many approaches that provide more or less effective solutions. As this paper deals with the idea of improving anomaly detection with SOMs, advantages and disadvantages of different SOM improvements will be given after briefly presenting the state of art regarding anomaly detection.

2.1. Anomaly Detection

The known approaches for anomaly detection are designing either misuse cases or attacks trees, as well as vari-
ous modification of the negative selection algorithm (NSA). Common to their construction is the necessity to think about which possible actions an intruder could do to achieve its goals, as well as to define adequate countermeasures. The drawback of having to think in advance about possible malicious actions severely limits their applicability.

A difficulty to predict potential attacks has resulted in the development of various NSA, which first map non-anomaly behaviors to the so-called self-states, and then generate detectors (antibodies) that cover everything else [7]. A decision that something is an anomaly is made always when being inside the coverage of a created detector. Because of its complexity this detection is too slow for applying to a real system.

A similar effect can be achieved by developing a statistical model of all self-states, which can also provide a probability that something is self-state or an anomaly [20].

However many of these solutions for anomaly detection are not fast and flexible enough to satisfy the practical requirements of building a system for intrusion detection. Since SOM can greatly help to meet these requirements, many known hybrid models are combining SOM, being a classification algorithm, and the NSA [11]. The essential advantage of using SOM for anomaly detection is its tolerance with the content of feature values. For instance it is allowed to mix Boolean values with time measures and click rates (as example for any old value). They just have to be normalized so that no bigger value takes advantage of any distance function. All IDS, including the SOM algorithm, have the drawback of classifying too much normal states as anomalies. Such alerts are called false positive.

2.2. SOM and its Improvements

The SOM is an unsupervised learning algorithm which is commonly used to visualize high-dimensional data on a two-dimensional screen [16] [17]. Because of its ability of projecting the high-dimensional spaces into the low-dimensional space, by revealing its underlying structure, SOM is nowadays used for anomaly detection, data mining, pattern recognition, vector quantization, image analysis, speech recognition, signal processing, medical applications, electronic-circuit design, robotics, linguistics and many other applications.

The advantage of SOM to other types of Artificial Neural Networks is their unsupervised aspect of not depending on knowing the input space. In spite of its high popularity, SOM encounters several drawbacks, such as being computationally expensive, needing much learning time, being memory lavishing, and searching slowly. These disadvantages have motivated the development of various improvements, each having its own advantages and disadvantages.

One of the first revisions of the original algorithm was the hierarchical SOM (HSOM) [4]. HSOM differs from the original SOM algorithm in just one step, saying that when there are more vectors mapped to a map unit after training than defined, these vectors are taken to train a new SOM. This helps being faster, because vectors do not have to be presented to all existing map units of the whole SOM during training. Being hierarchically can be seen as additional feature which improves every SOM. Nevertheless, a new parameter, determining the threshold of creating a child SOM, has to be defined.

Instead of growing hierarchically, Growing Grid SOM (GGSOM) grows by width and height. This grid is initialized with the size of $2 \times 2$. After some training iterations, two map units are searched: the most important unit and its neighboring least important unit. Then a row or column between these units, depending on their topological relation, is inserted [9]. In that way, the size of that SOM has not to be predefined, but the map might grow unshapely which is a disadvantage for the visualization.

Making the Growing Grid SOM hierarchical is in general, the idea of the Growing Hierarchical Self-Organizing Map (GHSOM), which is a hybrid of HSOM and GGSOM [19]. While GHSOM manages to speed up training, it inherits the disadvantage of GGSOM concerning its limited visualization.

The Growing Neural Gas (GNG) can be seen as a generalization of the SOM [8]. In the growing cell structure of the GNG, cells are similar to map units. They are created or pruned in dependence of their need to the neural net. The known grid neighborhood relationship is abolished and a free structure is introduced. This leads to the lack of being able to visualize the structure properly. However, the main advantage is that only needed map units are kept in memory and are used for computation.

Like GHSOM is hierarchical GGSOM, there is a hierarchical GNG. The Dynamic Adaptive Self-Organizing Hybrid Model (DASH) is a GNG, which expands hierarchically if there are too many data items mapped to one node. So the advantages of the GHSOM arise, which means that GNGs of deeper levels do not have to be trained with the whole data item collection [13].

The Scalable Self-Organizing Map (SSOM) improves the original SOM whenever the representing vectors are very sparse. Whenever these vectors contain only a few non-zero elements, this solution will speed up the updating of weight vectors and the computation of distances enormously. The algorithm for this feature is based on the original algorithm of Kohonen and delivers the same results. It was mathematically transformed to ignore zero elements and takes almost $O(S)$ time instead of $O(S^2)$, where S is the size of the collection [21].

The original SOM algorithm can be converted very easily to a torus SOM by connecting the opposite edges of the
map [14]. Thus the size of the neighborhood of each map unit is the same and mapping to the corners of the map will be avoided. In that way one can guarantee that clustering was not caused by a smaller neighborhood in the corners. Because this feature has no real drawback it should always be applied when using SOM.

For a smarter search within the map along with faster training, the search can be improved by starting at this point of the SOM where the best matching unit (BMU) will most likely be mapped. This fast winner search looks around the neighborhood of the old BMU to find the new one [15] [12]. Note that when using a torus SOM, the neighborhood around the BMU does not end on the edge of the map.

3. Problem Description

Most of modern companies exchange the information mainly based on the cooperation of computer systems. Such companies are unfortunately often under attack of intruders, who are for instance building Trojan horses in order to collect confidential information. These Trojan horses are usually specially built for the computer system of a particular company, and consequently they will not be detected by the installed signature-based anti-virus software. Obviously, the anti-virus software is not enough by itself, and the fight against such sophisticated intruders requires much better treatment.

The injected Trojan horses will sooner or later have to start collecting data with the intention to transfer them outside the network afterwards. The potential intrusions can be hopefully observed by monitoring both network traffic and usage rates of resources. In other words, a log of response time, volume transfers, and other activities of network nodes can protocol potential attacks. These measures can be taken for every node from any single communication port by counting service calls (click-rates), number of payloads and sizes of packages. In addition, the involved IP-ranges and the usage of different storage media or processors can be gotten as well.

Unfortunately there is no way to survey thousands of features measured from hundreds of individual network nodes or components manually. Nor is it possible designing general misuse cases [3] or attack trees [22] for each network node by hand separately. Each node may also have different features and a very different importance to the network. Hence there is a necessity of individual solutions for each node and rules to decide which solution fits to a given number and type of measured features. Self-adjusting, scalable and reliable protection systems are desperately needed in this field.

Fig. 1 presents a cut-out of a big network. This cut-out illustrates the common elements of every company network, such as databases, servers, terminals, laptops, being all connected to each other. An intruder has managed by hacking to infect one server with a Trojan horse. The infected communication is represented with red communication lines, as well as red color is utilized for clearly marking the infected computer. A detection system which can overcome the problems of adapting itself to any given network node individually, still has to challenge the huge amount of data which is collected. The handling of gained measures has to happen in real time without lavishing given resources like memory or processor. Alerts have to be given on time, otherwise intrusions are detected too late and the system is not protected well. The approach, which is going to be presented in this paper, will try to offer a foundation for addressing issues, such as autonomous detection of anomalies, self-adjusting on the type of measured data, scalability to the amount of data, reliability with the detection of anomalies, capability of real time processing, resource sparing working, and no manually readjusting of individual network nodes [6].

4. Approach

Since both the characteristics of different network nodes and the capabilities of known SOM algorithms vary greatly, the cornerstone of the given approach is in finding a appropriate SOM for every node. By mapping automatically and individually SOM types to different nodes, our proposed approach tries to exploit the advantages of each algorithm, while avoiding the application of those SOMs that do not suit to the data, being measured at the particular node.

The steps in Fig. 2 outline the process of anomaly detection based on the usage of multiple SOM algorithms. All steps are performed continuously. While Step 1 encapsulates a main contribution by reason of deciding which SOM will be mapped to each node, Steps 2 and 3 are more or less typical when SOM is used for anomaly detection [10]. Therefore, the ongoing discussions will focus on the build-
**Step 1: Data Acquisition**

**Step 1.1:** Analyzing and Adjusting Measures

**Step 1.2:** Mapping SOM to Network Node

**Step 2: Potential SOM Training**

**Step 3: Anomaly Detection based on SOM**

**Figure 2. Process of Anomaly Detection**

The blocks of Step 1, which first handle and transform the measured data (Step 1.1), and afterwards apply the experimentally found heuristic rules to decide which particular SOM will be utilized (Step 1.2).

Because of the great improving aspect of the SSOM (see Fig. 6(c) in the evaluation section), Step 1.1 is focused on reducing the number of non-zero elements. To transform vectors with the measured data to vectors containing very few non-zero elements, it is advisable to use the differences of the data to the preceding measures. The number of non-zero elements can be consequently reduced since some measured values might not have been changed.

Step 1 continues by specifying for every node the values of properties that are inputs for heuristic rules. While the number of zero elements and the number of features depend on the measured data that should be encapsulated into SOM, the importance of node and visualization requirements are set manually. The heuristic rules, which are obtained based on the evaluations given in the following section, are presented in Fig. 3.

The rules on Fig. 3 define the procedure of finding out which SOM is the most suitable for a particular node belonging to Step 1.2, having the known preferences concerning the visualization, the assigned importance for the network, and the found number of features that can be measured, whose dynamics define also the number of non-zero elements. These rules are sorted by importance and whenever two rules disagree, the more important rule is activated.

As the SSOM shows good behavior only when the number of zero elements dominates, Rule 1 ensures that SSOM will not be applied in situations where it can degrade performances. When saving memory this rule regards the changed limit noticed from Fig. 6(c). Fast winner search will always be applied by Rule 2, whenever the SOM should be fast to ensure real time processing. Rule 3 specifies that always when the number of features is smaller than 10, the original SOM should be utilized mostly because of its superior applicability when the visualization is needed and at the same time the underlying problem is not so difficult. Additionally Rule 3 is responsible for optimizing the usage of resources for big feature vectors by choosing HSOM for very economic training. Rule 4 handles the known groups from Fig. 6(b) combined with observations from Fig. 6(a). GHSOM variants loaded the processor just for a short time while needing more memory in contrast to the memory saving group seen in Fig. 6(b). Because the reliability of anomaly detection increases together with using more neurons for representing the input data, Rule 5 takes into account the importance of nodes by selecting the big SOMs with many neurons for important nodes. Finally Rule 6 considers the visualization when it is rarely needed.

As various SOM algorithms can be combined even in the single node, where GHSOM can utilize the principles behind SSOM, the rules define which SOM types can have the acceptable performances in given conditions. If for instance the importance of a node is medium (Rule 5) and at the same time the number of features is larger than 500 (Rule 3), the GHSOM is chosen. Such a decision is made since Rule 5 points out to GGSOM and Rule 3 selects HSOM, whose combination leads to GHSOM. In the case where additionally the number of non-zero elements is larger than 40%, the Rule 1 will be activated, and consequently the SSOM principles will be plugged into the GHSOM.

**Figure 3. Heuristic Rules**

| Rule 1: if the number of zero elements > 40% and no memory limitation then SSOM else if number of zero elements > 50% and memory limitation then SSOM with compressed vectors |
| Rule 2: if real time processing then fast winner search |
| Rule 3: if the number of features ≤ 10 then original SOM else if the number of features > 500 then HSOM |
| Rule 4: if limited memory resources and no processor limitation then HSOM, DASH or GNG else if limited processor resources and no memory limitation then GHSOM |
| Rule 5: if the importance of node is medium then GGSOM else if importance of node is high then big original SOM |
| Rule 6: if visualization is needed rarely then HSOM |

**Figure 4. Process of Anomaly Detection**

**Potential SOM Training**

**Data Acquisition**

**Mapping SOM to Network Node**

**Analyzing and Adjusting Measures**

**Anomaly Detection based on SOM**

**SSOM**

**HSOM**

**GHSOM**

**GGSOM**

**DASH**

**GNG**

**ORIGINAL SOM**

**BIG ORIGINAL SOM**

**SSOM WITH COMPRESSED VECTORS**

**FAST WINNER SEARCH**

**no memory limitation**

**memory limitation**

**limited processor resources**

**limited memory resources**

**no memory limitation**

**memory limitation**

**Figure 1. Original SOM Training**

**Figure 2. Process of Anomaly Detection**

**Figure 3. Heuristic Rules**

**Figure 4. Process of Anomaly Detection**
to detect anomalies in every network node.

In Step 2 the training or retraining of the SOM is decided. If so, the training can be performed in parallel with Step 3, not to disturb the continuously ongoing anomaly detection. From time to time the SOM has to be retrained to fit possible changed circumstances and to keep the trained SOM up to date. Such retraining can be forced by default or by analyzing the incoming measures. If these measures lead to a different type of SOM (see rules in Fig. 3), the actually chosen SOM will be replaced. It will be replaced as well, when the density of mapped measures in one point of the map is too high, because it points to the fact that the SOM is not representative enough anymore. The latest logged measures will be the base of such retraining.

Step 3 handles the anomaly detection and utilizes nearly the same detection technique as in [10]. For better establishment of clusters the training is extended with the training of the positive and negative squared values of the incoming measures. That leads to a bigger non-self region on the map which can be seen as gray shaded area in Fig. 5.

These steps ensure to meet the formulated requirements better than the use of the simple SOM algorithm. The obtained approach is effective, because of the autonomous detection of anomalies; accommodating, because of the self-adjusting on the type of measured data; scalable to the amount of data; safe, because it is reliable with the detection of anomalies; fast, because the improved SOMs are capable of real time processing; efficient, because of their resource sparing working; easy to use, because there is no need of readjusting the individual network nodes manually; adaptable, because even chosen SOMs can be changed when they do not meet the requirements anymore; robust, because the system is protected even when one node is not protected (e.g. node might be shut down); realistic, because the SOM is able to train every data which can be expressed in values.

5. Evaluation

In order to formulate heuristic rules that decide which SOM matches best under given circumstances, the evaluations are made to uncover the advantages and disadvantages of different SOM algorithms. Since the number of features plays the important role when the reliability of anomaly detection is concerned, different SOM algorithms are compared in time and in needed memory domains, regarding the number of features. Furthermore, because of the possibility to plug SSOM in any other SOM algorithm, the performances of SSOM will be inspected closer, regarding the percentage of zero elements. Finally, the ability of SOM to cluster anomaly and normal states will be illustrated.

The comparisons are made fair by basing the evaluations on a Java based implementation where several SOM algorithms can be combined to create different types of SOMs. Consequently, the finally obtained different SOMs are using the same core methods, which result that the noticed differences are purely the result of the encapsulated different logic.

As it can be seen in Fig. 6(a), there are no changes in the rankings of the fastest SOMs, whether small or huge feature vectors are taken. The fastest SOM is a GHSOM upgraded with the scalable feature and the fast winner search, followed by the scalable GHSOM and the SSOM. The Fig. 6(a) shows too, that the training time of a SOM which is just improved with the fast winner search is the fastest non-scalable SOM. GNG and DASH show no good results under the aspect training time and even the original SOM was faster.

The number of map units reveals strengths and weaknesses from another point of view. This feature directly points to the use of memory. Map units are the consisting parts of each SOM. More units mean more use of memory. Fig. 6(b) manifest three different classes of memory usage. In the first group of big memory use, all types of GHSOM can be found. In the second group are the SOMs which do not change their size. The last group of little memory usage consists of HSOM, DASH and GNG.

Fig. 6(c) shows a comparison between the original SOM and the SSOM algorithm. It can be seen that the SSOM takes more training time as long as there are approximately more non-zero elements than 40% in the trained vectors.
Because it is very resource lavishing to carry all zero elements in the feature vectors, which are not even needed in this algorithm, we additionally compared these algorithms with a variant which is trained with compressed vectors, which only carry non-zero elements. Here the improvement is under the limit of 50% non-zero elements. This variant is slower than SSOM, because elements in the feature vector can not be accessed directly with its original belonging index. Both variants have in common, that whenever the vectors are very sparse, the training time is significantly lower than in the original algorithm.

The simulation of the normal states of a network node is represented on Fig. 5. The areas of the normal, corresponding to self states, can be seen as a cluster which has developed by mapping incoming measures of three network nodes (represented with three colors). Because of the torus feature (see Section 2) this cluster reaches to the left side of this map too. The size of the circles points to the density of mapped measures to the map units. The gray region can be seen as region of abnormal states. Whenever a value changes in an abnormal way, it will be mapped to the gray area and will be marked as possible anomaly in dependence to the topological distance to the cluster. The structure of the SOM makes it possible to alert anomalies along with a percentage of being an anomaly by analyzing this distance combined with the distance to the last mapped location of this state. Because each value is predictable, an anomaly can be detected as well when the value is mapped to a self area, but too far away from its old area.

6. Future Work

The major task which will be performed in the course of future evaluations, is to concentrate on proving the usefulness of each formulated rule. Therefore a simulation is planned to practically demonstrate the ability of the presented approach to support large networks with better than the exclusive usage of the original SOM with available resources.

While the delicate decisions of which SOM algorithm to use for which network nodes are currently made by using manually generated and sometimes hard to maintain heuristic rules, the cornerstone of ongoing efforts will be concerned in making a system to be more autonomous. As every node has its own properties, such as the dynamics of changing the measured parameters, the importance to overall network, the number of sensed features, each SOM algorithm has also its unique capabilities towards supporting these different nodes. The underlying idea is in applying the similar reasoning for learning which SOM is best for which types of network nodes, as an approach from [2] uses to select the most appropriate filtering strategy based on the properties of a request that should be processed.

The yet another possible improvement is related with optimally using available system resources by learning about the different requirements of every SOM. The motivation for such a trial is found in the set of approaches [1], which all improve themselves by choosing the responsible filtering strategy through applying proportional selection, being excellent for exploiting the already learned properties of strategies, while at the same time exploring their unknown capabilities.

Additionally we will enhance the collection of SOMs by improving algorithms like the Spatial Access Methods Self-Organizing Map [5]. The belonging algorithm improves the winner search by creating a search tree upon the SOM, which directly leads to the weight vectors of the different map units.

The contradiction between accommodating the SOM to the input space completely and still having regions left for non-input which is now solved by additionally training the positive and negative squared values of the incoming measures, has to be investigated in detail.
7. Conclusion

The presented approach offers the capability to support the anomaly detection, by providing the ability of using particularized SOMs while still being able to handle real time requirements. Our approach meets the requirements for adequate anomaly detection by separately installing a different and individual SOM on every network node which has to be monitored. It can efficiently handle the underlying amount of data that should be observed, as well as automatically support the additional requirements regarding the needed visualization and the overall importance of a given node. Even though all these requirements are currently supported through the manually generated heuristic rules, authors’ hope is that the realized system provides a solid foundation, which can be easily further upgraded to become even more autonomous and intelligent.

While focusing on the adaption of SOM based anomaly detection to individual given conditions, this approach does improve the quality of the anomaly detection itself by providing the use of bigger, faster and more detailed SOMs.

Acknowledgment

Authors gratefully would like to thank to Robert Wetzker for providing insights into intrusion detection systems. Special thanks also go to Brijnesh Johannes Jain for hour long discussions that have strengthen the article.

References


