On the Capability of an SOM based Intrusion Detection System

H. Günes Kayacık, A. Nur Zincir-Heywood, Malcolm I. Heywood
Dalhousie University,
Faculty of Computer Science,
6050 University Avenue, Halifax, Nova Scotia. B3H 1W5

Abstract—**An approach to network intrusion detection is investigated, based purely on a hierarchy of Self-Organizing Feature Maps. Our principle interest is to establish just how far such an approach can be taken in practice. To do so, the KDD benchmark dataset from the International Knowledge Discovery and Data Mining Tools Competition is employed. This supplies a connection-based description of a factitious computer network in which each connection is described in terms of 41 features. Unlike previous approaches, only 6 of the most basic features are employed. The resulting system is capable of detection (false positive) rates of 89% (4.6%), where this is at least as good as the alternative data-mining approaches that require all 41 features.**

Index terms—**Intrusion Detection Systems, Self-Organizing Feature Map.**

I. Introduction

The Internet, as well as representing a revolution in the ability to exchange and communicate information, has also provided greater opportunity for disruption and sabotage of data previously considered secure. The study of systems able to detect network borne intrusions provides many challenges. Classical network based approaches to this problem often rely on either rule-based misuse detection or anomaly detection [1]. Rule-based misuse detection systems attempt to recognize specific behaviors that represent known forms of abuse or intrusion. On the other hand, anomaly detection attempts to recognize abnormal user behavior. Both approaches have their respective advantages and disadvantages. Rule based systems typically require an exhaustive list of templates characterizing each attack instance; there is no concept of similarity to a currently listed attack instance. The anomaly detection approach will actually identify "normal" behaviors by mining the monitored behavior of each user so that "abnormal" behaviors can be characterized. Clear distinctions between normal and abnormal, however, are difficult to achieve in practice.

Given the significance of the intrusion detection problem, there have been various initiatives that attempt to quantify the current state of the art. In particular the International Knowledge Discovery and Data Mining Tools Competition [2] provided the KDD-99 data set for assessing different AI approaches to the problem. Although not without its drawbacks [3], this benchmark provides the only *labeled* dataset for comparing IDS systems, which the authors are aware of. The most recent works in this area are able to provide detection (false positive) rates in the range of 91% (8%) to 98% (10%) whilst using all 41-connection features [4, 5].

In this work, we are interested in establishing how far an approach based on a sequence of hierarchical topological maps can be taken, whilst only utilizing a sub-set of the available 41-connection features. Specifically, the work only uses the six "Basic features of an Individual TCP connection" [2]. Six Self-Organizing Feature Maps (SOM) are then built, one for each input feature. The second level of the hierarchy integrates the information from each SOM and a third layer is selectively built for second layer neurons that respond to both attack and normal connections. Neurons in the second and third layers are therefore labeled using the training set, but the training process itself is entirely unsupervised. Detection (false positive) rate of the detector on the test set varies between 89% (4.6%) to 99.7% (1.7%) depending on the KDD-99 test partition employed.

The remainder of the paper is organized as follows. Section II provides the background and methodology of the work. Details of each learning algorithm comprising the system are given in Section III. Results are reported in Section IV and Conclusions drawn in Section V.

II. Methodology

As indicated in the introduction, the basic objective of this work is to assess how far a machine learning approach may be taken which makes minimalist use of any *a priori* domain knowledge. To this end, an approach based on topological maps is employed. This assumes that given sufficient resolution in the maps, it is possible to separate normal from attack behavior. In a previous work, we established that by utilizing a shift register to embed the temporal relationship between incoming connections, described in terms of session information, a simple two layer SOM hierarchy was sufficient to distinguish between different behaviors [6]. However, the dataset used in that scheme only consisted of seven attacks. In this work, on the other hand, we thoroughly benchmark the hierarchical SOM methodology on the KDD-99 benchmark. To this end, we first describe the characteristics of the data set and then the SOM architecture utilized.

1) *KDD dataset:*

The KDD-99 dataset is based on the 1998 DARPA initiative to provide designers of intrusion detection systems (IDS) with a benchmark on which to evaluate different methodologies [7]. To do so, a simulation is made of a factitious military network consisting of three 'target' machines running various operating systems and services.

This work was supported in part by Discovery grants, of the second and third authors, from the Natural Sciences and Engineering Research Council of Canada.

Additional three machines are then used to spoof different IP addresses, thus generating traffic between different IP addresses. Finally, there is a sniffer that records all network traffic using the TCP dump format. The total simulated period is seven weeks. Normal connections are created to profile that expected in a military network and attacks fall into one of five categories: User to Root; Remote to Local; Denial of Service; Data; and Probe. Note that User to Root and Remote to Local can represent content-based attacks, and may therefore only be detected indirectly by the type of system developed in this work (e.g. guessing passwords often manifests itself as multiple attempted login's between the same source destination pair).

In 1999 the original TCP dump files were preprocessed for utilization in the Intrusion Detection System benchmark of the International Knowledge Discovery and Data Mining Tools Competition [2]. To do so, packet information in the TCP dump file are summarized into connections. Specifically, "a connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol" [8]. This process is completed using the Bro IDS [9], resulting in nine "Basic features of an Individual TCP connection" [8]; hereafter referred to as 'basic features',

- Duration of the connection;
- Protocol type, such as TCP, UDP or ICMP;
- Service type, such as FTP, HTTP, Telnet;
- Status flag, derived by Bro to describe a connection;
- Total bytes sent to destination host;
- Total bytes sent to source host;
- Whether source and destination addresses are the same or not;
- Number of wrong fragments;
- Number of urgent packets;

Note that only Protocol and Service features are not derived i.e. they are estimated immediately as opposed to after a connection has completed. Moreover, the above 'status flag' should not be confused with the TCP/IP suit flags. Finally, last three features are specific to certain attack types (no variation is observed across the normal data in the training set), hence these terms were ignored in this work.

In addition to the above nine 'basic features,' each connection is also described in terms of an additional 32 *derived* features, falling into three categories,

- *Content Features*: Domain knowledge is used to assess the payload of the original TCP packets. This includes features such as the number of failed login attempts;
- *Time-based Traffic Features*: These features are designed to capture properties that mature over a 2 second temporal window. One example of such a feature would be the number of connections to the same host over the 2 second interval;
- *Host-based Traffic Features*: Utilize a historical window estimated over the number of connections – in this case 100 – instead of time. Host based features are therefore designed to assess attacks, which span intervals longer than 2 seconds.

In this work, none of these additional features are employed.

2) *Hierarchical SOM:*

As in our earlier work, a hierarchical SOM architecture is employed [6]. Our basic motivation is to steadily build more abstract features as the number of SOM layers increase. That is to say, our hypothesis is that features learnt at the initial layers of a hierarchy may still be interpreted in terms of recognizable basic measured properties, whereas features at the highest level in the architecture will capture aspects synonymous with normal or attack behaviors. Specifically, three layers are employed. In the first, individual SOMs are associated with each basic TCP feature. This provides a concise summary of the interesting properties of each basic feature, as derived over a suitable temporal horizon. The second layer integrates the views provided by the first level SOMs into a single view of the problem. At this point, we use the training set labels associated with each pattern to label the respective best matching unit in the second layer. The third and final layer is built for those neurons, which win for both attack and normal behaviors. This results in third layer SOMs being associated with specific neurons in the second layer. Moreover, the hierarchical nature of the architecture means that the first layer may be trained in parallel and the third layer SOMs are only trained over a small fraction of the data set.

3) *Preprocessing and Clustering*

In order to build the hierarchical SOM architecture, several data normalization operations are necessary, where these are for the purposes of preprocessing and inter-layer 'quantization' of maps. Preprocessing has two basic functions, to provide a suitable representation for the initial data and support the representation of time. In the case of initial data representation, three of the basic features – Protocol type, Service type and Status flag – are alphanumeric. As the first SOM layer treats each feature independently, we merely map each instance of an alphanumeric character to sequential integer values. Numerical features – connection duration, total bytes set to destination/ source host – are used unchanged.

In the case of representing time, the standard SOM used here has no capacity to recall histories of patterns directly. However, sequence as opposed to time stamp, is the property of significance in this work [6]. A shift register of length 'l' is therefore employed in which a 'tap' is taken at a predetermined repeating interval 't' such that $l \% t = 0$, where % is the modulus operator. The first level SOMs only receive values from the shift register that correspond to tap locations. Thus, as each new connection is encountered (enters at the left), the content of each shift register location is transferred one location (to the right), with the previous item in the lth location being lost.

The requirement for 'quantization' occurs between the first and second level SOMs. Specifically, the purpose of the second level SOM is to provide an integrated view of the input feature specific SOMs developed in the first layer. There is therefore the potential for each neuron in the second layer SOM to have an input dimension defined by the total neuron count across all first layer SOM networks. This would be a brute force solution that does not scale computationally (there are half a million training set patterns). Moreover, given the topological ordering provided by the SOM, neighboring neurons will respond to similar stimuli. We therefore quantize the topology of each first layer SOM in terms of a fixed number of neurons and re-express the first layer best matching units in terms of these. This significantly reduces the dimension seen by neurons in the second layer SOM.

III. Learning Algorithms

Two learning algorithms are used to build the hierarchical SOM architecture. The first is used to train each SOM in the hierarchy [10]. The second is a clustering algorithm that is used to quantize the number of SOM neurons 'perceived' by the second layer. In the latter case the Potential Function algorithm is employed [11]. These are briefly summarized as follows.

A. Self-Organizing Feature Map

Kohonen's Self-Organizing Feature Map (SOM) algorithm is an unsupervised learning algorithm in which an initially 'soft' competition takes place between neurons to provide a topological arrangement between neurons at convergence [10]. The learning process is summarized as follows,

1. Assign random values to the network weights, w_{ij};
2. Present an input pattern, $\boldsymbol{x}$, in this case a series of taps taken from the shift register providing the 'reconstruction' state space on which the SOM is to provide a suitable quantized approximation.
3. Calculate the distance between pattern, $\boldsymbol{x}$, and each neuron weight $\boldsymbol{w}_j$, and therefore identify the winning neuron, or

$$d = \min_j \left\{ \left\| x \square w_j \right\| \right\} \quad (1)$$

   where $\|\cdot\|$ is the Euclidean norm and $\boldsymbol{w}_j$ is the weight vector of neuron j;
4. Adjust all weights in the neighborhood of the winning neuron, or

$$w_{ij}(t+1) = w_{ij}(t) + \square(t)K(j,t)\left\{ x_i(t) \square w_{ij}(t) \right\} \quad (2)$$

   where $\square(t)$ is the learning rate at epoch t; and $K(j, t)$ is a suitable neighborhood function;
5. Repeat steps (2) – (4) until the convergence criterion is satisfied.

Following convergence, presentation of an input vector, $\boldsymbol{x}$, results in a corresponding output vector, $\boldsymbol{d}$, the Euclidian distance between each neuron and input. The neuron with the smallest distance represents the winning or best matching neuron, step (3). The best matching neuron also defines a neighborhood of next nearest neighboring neurons. Once the maps are trained, it is this concept of a best matching node that is used to facilitate the labeling of the second and third level maps.

B. Potential Function Clustering

The Potential Function Clustering algorithm consists of four steps [11],

1. Identify the potential of each data point relative to all other data points. All data points represent candidate cluster centers;
2. Select the data point with largest potential and label as a cluster center;
3. Subtract the potential of the data point identified at step (2) from all others and remove this point from the list of candidate cluster centers;
4. Repeat on step (2) until the end criterion is satisfied.

In this application, the set of data points correspond to the set of neurons in each (first layer) SOM, where the weights of each neuron describe a neuron position in terms of the original input space. Step 1 characterizes neurons in terms of how close they are to others. A neuron with many local neighbors should have a high 'potential' as expressed by a suitable cost function, or

$$P_t(w(j)) = \overset{M}{\underset{i=1}{\square}} \exp\left(\square\square \left\| w(i) \square w(j) \right\|^2 \right)$$

where $w(j)$ is the 'j'th SOM neuron, $P_t(w(j))$ is the potential for such neuron at iteration t, M is the number of data points (in this case SOM neurons), and $\square$ is the cluster radii.

Step 2 identifies a candidate cluster center (SOM neuron) by choosing the point with largest potential $P_t(x^*)$. Step 3 removes the influence of the chosen neuron from the remaining (unselected) set of SOM neurons. That is, the remaining neurons have their respective potentials decreased by a factor proportional to the distance from the current cluster center, or

$$P_{t+1}(w(j)) = P_t(w(j)) \\ \square P_t(w^*) \exp\left(\square \square \left\| w(i) \square w(j) \right\|^2 \right)$$

where $t + 1$ is the index of the updated potential at iteration t; w^* is the data point associated with the current cluster center, and $\square$ is the cluster radii ($\square < \square$).

The result of step 3 is the labeling of a specific SOM neuron as a cluster center. Step 4 iterates the process in conjunction with some suitable stop criterion. In this case, we stop when six cluster centers are identified, where the alpha and beta values are set accordingly. That is to say, further cluster centers correspond to points with a potential value less than 10% of the first potential located. The net effect of this process is therefore that each of the six first layer SOMs are characterized in terms of 6 clusters, resulting in a total of 36 inputs to the second level SOM.

Once the 6 cluster centers are identified for each SOM, representing the 'quantized' SOM output, we normalize as follows,

$$y = \frac{1}{1 + \left\| w \square x \right\|}$$

where w is the cluster center and $\boldsymbol{x}$ is the original SOM input.

The second layer SOM now receives a vector, ***y***, of the form,

$$\boldsymbol{y} = [y_{1,1}, \ldots, y_{1,6}, y_{2,1}, \ldots, y_{i,j}]$$

where i is the SOM index and j is the cluster (neuron) index.

IV. RESULTS

In all cases the SOM Toolbox and SOM-PAK were employed for the design of each SOM comprising the SOM hierarchy [12]. In the following we describe the dataset, training procedure and evaluation of the proposed architecture.

A. *KDD-99 Dataset*

The KDD-99 data consists of several components, Table I. As in the case of the International Knowledge Discovery and Data Mining Tools Competition, only the '10% KDD' data is employed for the purposes of training [2]. This contains 24 attack types and is essentially a more concise version of the 'Whole KDD' dataset. One side effect of this, is that it actually contains more examples of attacks than normal connections. Moreover, the attack types are not represented equally, with Denial-of-Service attack types – by the very nature of the attack type – accounting for the majority of the attack instances. However, both test sets contain an additional 14 (unseen) attacks. The so-called 'Corrected (Test)' dataset provides a dataset with a significantly different statistical distribution than either '10% KDD' or 'Corrected (Test)'.

TABLE I
BASIC CHARACTERISTICS OF THE KDD DATASET

Dataset label	Total Attack	Total Normal
10% KDD	396,744	97,277
Corrected (Test)	250,436	60,593
Whole KDD	3,925,651	972,780

B. *Training*

Learning parameters for the SOMs are summarized in Table II, where this process is repeated for each SOM comprising the hierarchy. In each case, training is completed in two stages, the first providing for the general organization of the SOM and the second for the fine-tuning of neurons. Table III summarizes the additional parameters utilized by the shift register and Potential Function clustering algorithm. The resulting SOM hierarchy consists of 6 SOM networks in the first layer (temporal encoding), each consisting of 6□6 grid and 20 inputs. Potential Function clustering 'quantizes' each original first layer SOM to six neurons using the process described in Section III.B, resulting in 36 inputs to the second layer SOM (responsible for integration). Once training of the second layer is complete, labeling takes place. That is, for each connection in the training set, the corresponding label is given to the best matching unit in the second layer. A count is kept for the number of normal and attack connections each best matching unit receives. Third layer SOMs are built for second layer SOM neurons that demonstrate significant counts for both attack and normal connections, Section IV.C. This results in 6 third layer SOMs being built on top of specific second layer neurons. In each case third layer SOMs consist of 20 □ 20 neurons, where a larger neuron count is utilized in the third layer in order to increase the likelihood of separation between the two connection types. Moreover, only connections for which the corresponding second layer SOM is the best matching unit are used to train third layer SOMs, facilitating the use of larger SOMs without experiencing a high computational overhead. Finally, in each case, the inputs to the third level SOMs correspond to the 36-element vector of 'quantized' first layer outputs.

TABLE II
SOM TRAINING PARAMETERS

Parameter	Rough Training	Fine Tuning
Initial □	0.5	0.05
□ decay scheme	$f(\text{epoch}^{-1})$	
Epoch Limit	4,000	
Neighborhood Parameters		
Initial Size	2	1
Function	Gaussian	
Relation	Hexagonal	

TABLE III
POTENTIAL FUNCTION AND SHIFT REGISTER PARAMETERS

	Potential Function (For each feature respectively)	Shift Register	
▯	e-6, e-3, 2e-7, e-6, 16e-7, 0.1599015,	Length	96
▯	2e-2, 2e-2, e-2, 4e-2, e-1, e-2	# Taps	20

C. *Evaluation*

Performance of the classifier is evaluated in terms of the false positive and detection rates, estimated as follows,

$$\text{Skipped Rate} = \frac{\text{Number of Skipped Connections}}{\text{Total number of Connections}}$$

$$\text{False Positive Rate} = \frac{\text{Number of False Positives}}{\text{Total number of Normal Connections}}$$

$$\text{Detection Rate} = 1 - \frac{\text{Number of False Negatives}}{\text{Total number of Attack Connections}}$$

Where False Positive (Negative) Rate is the number of Normal (Attack) connections labeled as Attack (Normal).

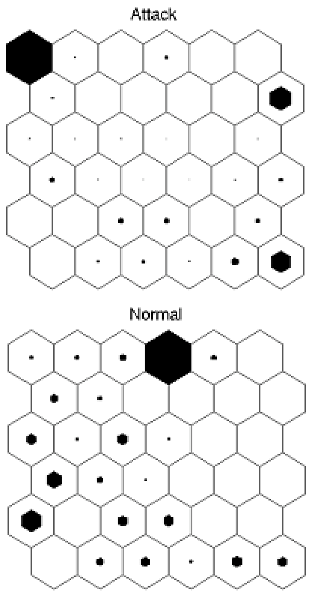

Fig. 1. Hit histogram of the second level map.

Figure 1 summarizes the count of attack and normal connections in the second level SOM; where proportionally larger counts result a greater area of the hexagon being colored. It is apparent that nodes 1, 32 and 36 account for most of the attack connections and neuron 19 most of the normal connections. It is also apparent that several neurons also respond to both normal and attack connections. To this end neurons 4, 17, 18, 23, 30 and 36 are selected for association with third level SOMs, one for each second layer neuron, where Table IV details the respective counts of normal and attack connections.

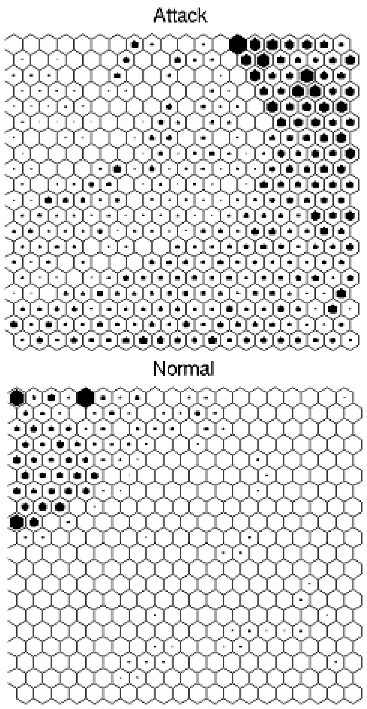

Figure 2: Third level map on top of second level map's neuron 36

On completion of training at the third level maps, a clear separation between normal and attack was achieved. Figure 2 illustrates this property for the case of neuron 36. In this case, it is clear that the normal connections all reside in the top left corner of the map, whereas the attack connections populate the remainder. Finally, the larger SOMs utilized in the third layer could result in neurons that remain unlabeled. These are listed as 'skipped' in the following analysis of test set performance.

TABLE IV
COUNT OF ATTACK AND NORMAL CONNECTIONS PER 2ND LAYER CANDIDATE NEURON

Neuron	Normal	Attack
4	2,177	2,613
17	2,051	3,151
18	1,731	1,706
23	2,304	3,204
30	2,453	5,292
36	1,688	45,440

TABLE V
TEST SET RESULTS

Network level	Skipped	FP rate	Detection rate
Corrected (Test)			
Level 2	0 %	7.6 %	90.6 %
Level 3	0.7 %	4.6 %	89 %
Whole KDD			
Level 3	0.06 %	1.7 %	99.7%

Table V details test set performance for the case of a two layer and three layer hierarchy on the 'Corrected (Test)' KDD data set and the three-layer hierarchy on the 'whole KDD' test set. The superiority of the three-layer architecture is again demonstrated. Finally, performance of the two-layer and three-layer hierarchy on corrected test set for different categories is summarized in Table VI, whereas the performance on new attacks in 'Corrected Test' set is detailed in Table VII. It is readily apparent that as the hierarchical SOM approach is only utilizing 'basic features' (i.e. none of the content-based features are employed), performance on network-based attack categories (DoS, Probe) is naturally much better than that on content-based categories (U2R, R2L).

TABLE VI
DETECTION RATE OF NEW ATTACKS FOR 2-LAYER AND 3-LAYER HIERARCHY

	Normal	DoS	Probe	U2R	R2L
Level 2	92.4	96.5	72.8	22.9	11.3
Level 3	95.4	95.1	64.3	10.0	9.9

TABLE VII
DETECTION RATE OF NEW ATTACKS FOR 2-LAYER AND 3-LAYER HIERARCHY

Attack Name	Level 2	Level 3
Apache2.	90.3	90.7
httptunnel.	58.9	20.9
mailbomb.	7.8	6.8
mscan.	90.2	60.9
named.	23.5	0.0
processtable.	59.4	47.6
ps.	0.0	0.0
saint.	79.1	78.7
sendmail.	5.9	11.8
snmpgetattack.	11.5	10.3
udpstorm.	0.0	0.0
xlock.	0.0	0.0
xsnoop.	0.0	0.0
xterm.	23.1	30.8

TABLE VIII
RECENT RESULTS ON THE KDD BENCHMARK

Technique	Detection Rate	FP Rate
Data-Mining [5]	70-90%	2%
Clustering [4]	93%	10%
K-NN [4]	91%	8%
SVM [4]	98%	10%

Table VIII provides a summary of some recent results from alternative approaches trained on the KDD-99 dataset and tested using the 'Corrected (Test)' data [4, 5]. Detection rates are very similar to those reported for the SOM hierarchy constructed here. However, there are actually several additional factors with which these results need to be interpreted. Firstly, all the data mining approaches are based on all 41 features; the SOM hierarchy only utilizes 6. Secondly, in the case of [4], figures quoted are for a mixture of specific and multiple attack types, making it difficult to determine performance over the entire dataset. Thirdly, use was also made of Host based information, thus providing an advantage when detecting content based attacks [4].

V. CONCLUSION

A hierarchical SOM approach to the IDS problem is proposed and demonstrated on the International Knowledge Discovery and Data Mining Tools Competition intrusion detection benchmark [2]. Specific attention is given to the representation of connection sequence (time) and the hierarchical development of abstractions sufficient to permit direct labeling of SOM nodes with connection type. Other than these two concepts, no additional use of *a priori* information is employed. In comparison to data mining approaches currently proposed, the approach provides competitive performance whilst utilizing a fraction of the feature set (6 of the 9 "Basic features of an Individual TCP connection" and none of the 32 additional higher-level derived features).

It is anticipated that future work will investigate the utilization of such a scheme within the context of a distributed solution to the IDS problem.

REFERENCES

[1] T. Bass, "Intrusion detection systems and multisensor data fusion," *Communications of the ACM*, 43(4), pp. 99-105, April 2000.

[2] S. Hettich, S.D. Bay, *The UCI KDD Archive*. Irvine, CA: University of California, Department of Information and Computer Science, http://kdd.ics.uci.edu, 1999.

[3] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, 3(4), pp. 262-294, 2001.

[4] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security*, Chapter 4, D. Barbara and S. Jajodia (editors), Kluwer, ISBN 1-4020-7054-3, 2002.

[5] W. Lee, S. Stolfo, K. Mok, "A data mining framework for building intrusion detection models," *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pp. 120-132, 1999.

[6] Lichodzijewski P., Zincir-Heywood A.N., Heywood M.I., "Host-Based intrusion detection using Self-Organizing Maps," *IEEE International Joint Conference on Neural Networks*, pp. 1714-1719, May 12-17, 2002.

[7] *The 1998 intrusion detection off-line evaluation plan*. MIT Lincoln Lab., Information Systems Technology Group. http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt, 25 March 1998.

[8] *Knowledge discovery in databases DARPA archive*. Task Description. http://www.kdd.ics.uci.edu/databases/kddcup99/task.html

[9] *Bro user manual*, http://www.icir.org/vern/bro-manual/index.html

[10] T. Kohonen, *Self-Organizing Maps*. 3rd Ed., Springer-Verlag, ISBN 3-540-67921-9, 2000.

[11] S.L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent and Fuzzy Systems*, 2, pp. 267-278, 1994.

[12] *Software Packages from Helsinki University of Technology*, Laboratory of Computer and Information Science Neural Networks Research Centre, http://www.cis.hut.fi/research/software.shtml