A New System for Clustering and Classification of Intrusion Detection System Alerts Using Self-Organizing Maps

Amir Azimi Alasti Ahrabi amir.azimi.alasti@gmail.com
Department of Computer
Islamic Azad University, Shabestar Branch
Tabriz, East Azerbaijan, Iran

Ahmad Habibizad Navin ah_habibi@iaut.ac.ir
Department of Computer
Islamic Azad University, Science and Research Branch
Tabriz, East Azerbaijan, Iran

Hadi Bahrbegi hadi.bahrbegi@gmail.com
Department of Computer
Islamic Azad University, Shabestar Branch
Tabriz, East Azerbaijan, Iran

Mir Kamal Mirnia mirnia-kam@tabrizu.ac.ir
Department of Computer
Islamic Azad University, Science and Research Branch
Tabriz, East Azerbaijan, Iran

Mehdi Bahrbegi m.bahribayli@gmail.com
Department of Computer
Islamic Azad University, Shabestar Branch
Tabriz, East Azerbaijan, Iran

Elnaz Safarzadeh elnaz_safarzadeh@yahoo.com
Department of Computer
Islamic Azad University, Shabestar Branch
Tabriz, East Azerbaijan, Iran

Ali Ebrahimi ali.ebrahimi1781@gmail.com
Department of Computer
Islamic Azad University, Shabestar Branch
Tabriz, East Azerbaijan, Iran

Abstract

Intrusion Detection Systems (IDS) allow to protect systems used by organizations against threats that emerges network connectivity by increasing. The main drawbacks of IDS are the number of alerts generated and failing. By using Self-Organizing Map (SOM), a system is proposed to be able to classify IDS alerts and to reduce false positives alerts. Also some alert filtering and cluster merging algorithm are introduce to improve the accuracy of the proposed system. By the experimental results on DARPA KDD cup 98 the system is able to cluster and classify alerts and causes reducing false positive alerts considerably.

Keywords: IDS, alert clustering, SOM, false positive alert reduction, alert classification.

1. INTRODUCTION

An IDS is a device that monitors system and/or network activities for malicious activities or policy violations and produces alerts to a Management Station. IDSs are generally divided into two categories based on detection method: Signature based IDSs and anomaly based IDSs [1]. In IDS there are two major problems namely generating many alerts and high rate of false positive alerts. Alert management methods are used to manage with these problems. One of the methods of alert management is clustering the alerts. According to recent researches, clustering the alerts is an NP-Complete problem [21].

In this paper by using SOM the proposed system classifies and clusters the alerts and also detects false positive alerts. Two algorithms are used in this system to filter alerts to train the SOM better and to merge generated clusters to reduce the number of clusters depending on the types of the attacks. Moreover to obtain a better result from SOM a preprocessing process is applied to the alerts during train and test phases.

The subject is introduced briefly in Section 1. Section 2reviews related works, section 3 explains the suggested system for classifying and clustering the alerts, the experimental results are shown in section 4 and finally section 5 is a conclusion and feature works.

2. RELATED WORKS

K. Julisch [2] proposed a clustering technique witch based on forming a generalized view of false alerts. This technique is for discovering root causes of false positive alerts. Julisch notice that a small number of root causes implies 90% of alerts. by identifying and removing this root causes total number of alerts come down to 82%.

Cuppens in his Mirador project used an expert system for clustering. By this method the alerts are entered to database with XML format and then to decide whether these be merged into a cluster the expert system algorithm is used [4, 5]. Jianxin Wang, et al, have used genetic algorithm for clustering alerts from IDS [10]. Also two clustering algorithms, based on GA and IGA are compared together [11]. Wang applied GA and IGA instead of Julisch's algorithm for "root cause" clustering. To distinguish malicious traffic from normal traffic the SOM is used [7, 8]. Also it has been proved [7, 8] that SOM-based IDS can handle two situations by discovering new attacks. Hayoung, et al. used SOM for real time detection of attacks in IDSs [9].

Maheyzah Md Siraj, et al. compared the clustering algorithms, EM, SOM, K-means and FCM on Darpa 2000 dataset [3]. The results show that EM algorithm is the best for clustering. Since the alerts received by SOM are not filtered thus the result for the SOM could be in doubt.

The main features of the proposed system are obtaining results with high accuracy which is due to filtering alerts and merging the generated clusters and also to reduce the number of false positive alerts considerably.

3. CLUSTERING AND CLUSSIFICATION SYSTEM

The proposed system is shows in Figure 1. schematically. The system depends on produced alerts directly by IDS. To generate alerts, Snort tool [12] with Darpa 98 dataset [13] is used. Snort is an open source signature based IDS which gets Darpa 98 online traffic and then generates alert log files. The alert log files are used as the inputs of the system.

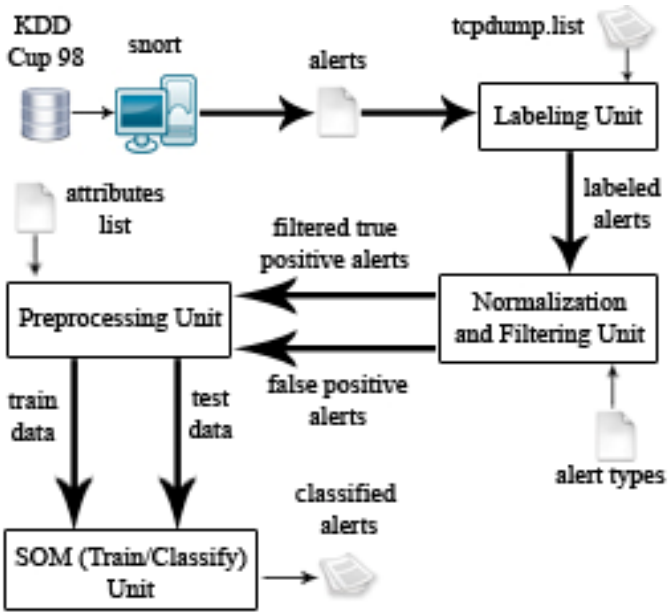

FIGURE 1: Alert clustering and classification system.

2.1 Labeling Unit

In Darpa 98 dataset, there are some tcpdump.list files for each online traffic flow which contains information of attacks. Labeling unit gets the alert and tcpdump.list files returning a list of labeled alerts involving type of attack for each alert.

An algorithm is proposed to generate map between alerts and attacks by using a unique key. This key consists source IP, destination IP, source port, destination port and ICMP code/type. The algorithm is shown in Figure 2.

1. Input TCPDUMP list files.
2. Input alert log files.
3. Create an empty AttackList set.
4. Create an empty AlertList set.
5. For each row in TCPDUMP list files:
   5.1. If the row is a labeled attack then add the row to the AttackList set.
6. For each row in alert log files:
   6.1.Create key with the five attributes: source ip, destination ip, source port, destination port, ICMP code/type.
   6.2. If the key exists in AttackList set then label the selected row with the type of found attack from AttackList set.
   Else
   Label the selected row with the False Positive attack type.
   6.3. Add the selected row to the AlertList set.
8. Return the AlertList set.

FIGURE 2: The algorithm of alerts labeling.

2.2 Normalization and Filtering Unit

Since Snort is a signature based IDS, it can't detect some of attacks like Pod and Smurf. It means that among the available attack type in Darpa 98 dataset, it can detect only eight cases with high accuracy [22]. So this unit takes the list of acceptable attacks, selected attributes and labeled alerts and then produces the list of filtered false and true positive alerts (Figure 1.). In normalization process eight attributes are chosen among the collection of alert attributes [14] stored in a vector named alert vector. The chosen attributes are: Signature ID, Signature Rev,

Source IP, Destination IP, Source Port, Destination Port, Datagram length and Protocol. One of the similar alerts based on values of the alert vector is selected in filtering process. Experiment shows that filtering the similar alerts wouldn't remove two alerts with two different types of attack.

2.3 Preprocessing Unit

Some attributes in the alert vector are string type and some numerical type. In this unit the string values are converted into numerical values and the range of the whole attributes is reduced. This unit takes the list of false positive and filtered true positive alerts and produces train and test data. (Figure 1)

By using (1) and (2) the string values are converted into numerical values.

$$IP = X_1.X_2.X_3.X_4, \qquad (1)$$
$$IP\_VAL = (((X_1 \times 255) + X_2) \times 255 + X_3) \times 255 + X_4$$

$$protocol\_val = \begin{cases} 0, protocol = None \\ 4, protocol = ICMP \\ 10, protocol = TCP \\ 17, protocol = UDP \end{cases} \qquad (2)$$

Since the differences in the range of the values will lose the accuracy of result, so the values of alert vector should be normalized. By using (3, 4), we convert [X_{min}, X_{max}] into [0, 1] (Unit Range) and into [0.1, 0.9] (Improved Unit Range).

$$UR = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad (3)$$

$$IUR = 0.8 \times \frac{x - x_{\min}}{x_{\max} - x_{\min}} + 0.1 \qquad (4)$$

2.4 SOM (Train/Classify) unit

- **SOM**

Self-organizing map [6] is a type of artificial neural networks that is trained using unsupervised learning. SOM describes a mapping from a higher dimensional input space to a lower dimensional map space. When a training example is fed to the network, its Euclidean distance to all weight vectors are computed (5). The neuron with weight vector which is similar to the input mostly is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM lattice are adjusted toward the input vector. The magnitude of the change decreases with time and with distance from the BMU.

$$\|x - m\|^2 = \sum_{k \in K} w_k |x_k - m_k|^2 \qquad (5)$$

where K is the set of known (not missing) variables of sample vector **x**, x_k and m_k are *kth* components of the sample and prototype vectors and w_k is the *k*th weight value. In this paper, the neighbour function

$$\exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) \qquad (6)$$

is Gaussian, where r_c is the location of unit c on the map grid and the $\sigma(t)$ is the neighbourhood radius at time t. And the learning algorithm

$$m_i(t+1) = \frac{\sum_{j=1}^{n} h_{ic(j)}(t) x_j}{\sum_{j=1}^{n} h_{ic(j)}(t)}, \quad (7)$$

is batch, where *c(j)* is the BMU of sample vector $\mathbf{x}_j$, $h_{i,c(j)}$ the neighbourhood function (the weighting factor), and *n* is the number of sample vectors.

- **Training the SOM**

Test data and train data are used as the input for this unit. For each feature, SOM makes the corresponding maps and then construct U-matrix (unified matrix) based on all feature maps [9]. U-matrix method allows to get a more suitable information of the vector distribution. This method is capable to classify all artificially generated data correctly [16]. The algorithm of SOM [17, 18, 19] is described in Figure 3 and can build U-matrix for normalized filtered alerts data in Figure 4. In U-matrix the lighter color neurons mean the borders of clusters.

Since feature maps and U-matrix obtained through two normalization methods (UR and IUR) are similar (the only difference is the range of corresponding feature maps and U-matrixes), thus diagrams of UR are shown.

1. Initialize the network.
   For each node *i* set the initial weight vector $w_i(0)$ to be random.
   Set the initial neighborhood $N_i(0)$ to be large.
2. Present the input.
   Present *x(t)*, the input pattern vector *x* at time *t* (*0<t<n* where *n* is the number of iterations defined by the user) to all nodes in the network simultaneously.
   x may be chosen at random or cyclically from the training data set.
   3. Calculate the winning node.
   Calculate node *c* with smallest distance between the weight vector and the input vector
   $$\|x(t) - w_c(t)\| = \min_i \{\|x(t) - w_i(t)\|\}$$

Hence

$$c = \arg\min \{\|x(t) - w_i(t)\|\}$$

4. Update the weights.
   Update weights for *c* and nodes with neighborhoods $N_c(t)$.
   $$m_i(t+1) = \frac{\sum_{j=1}^{n} h_{ic(j)}(t) x_j}{\sum_{j=1}^{n} h_{ic(j)}(t)},$$
   Where *c(j)* is the BMU of sample vector $\mathbf{x}_j$, *n* is the number of sample vectors and $h_{i,c(j)}$ the neighborhood function (the weighting factor)
   $$h_{ci}(t) = \alpha(t).\exp(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)})$$
5. Present the next input.
   Decrease h_{ci} so that $h_{ci}(t+1) < h_{ci}(t)$
   Reduce the neighborhood set so that $N_i(t+1) \subset N_{ci}(t)\ \forall_i$
   Repeat from step 2 choosing a new unique input vector $x(t+1) \neq x(j)$, $j \leq t$
   Until all iterations have been made ($t=n$).

FIGURE 3: SOM Algorithm to Construct Maps quoted from [17, 18, 19].

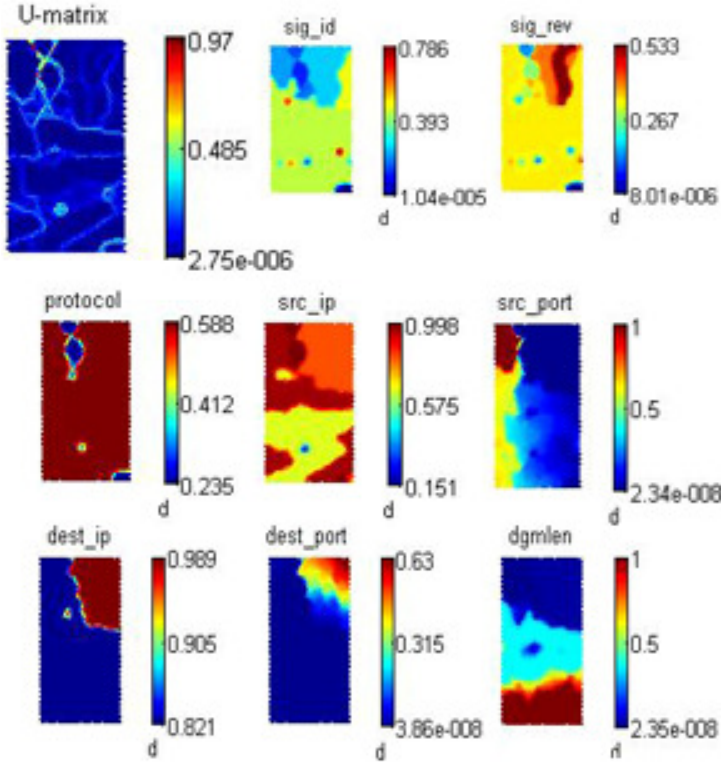

FIGURE 4: U-matrix and 8 Feature Maps.

- **Clustering of U-matrix**

For clustering of U-matrix, distance based U-matrix algorithm with 3 neuron neighbourhoods has been used [20] (Figure. 6. b). To label the clusters, the number of corresponding labels is calculated for each data vector in the cluster and the label with the higher density is supposed to be the label of the cluster. Because there is too many clusters, clusters merging algorithm is executed over clusters (Figure 5.). In this algorithm, the clusters with blank label are supposed to be unknown. Unknown clusters are the borders of various parts of U-matrix. Figure 6. c. shows merged clusters after execution of algorithm.

1. Input clustered U-matrix, *U*.
2. Creating of the initial clusters.
   Let *C* an empty set.
   For each attack type in *U*:
   Add $C_{AttackType}$ set to *C*.
3. Finding correspond clusters for each attack type in *U*.
   For each cluster in *U*:
   Let *SomeClusterOfU* = select a cluster from *U*.
   AttackType= find the name of the *SomeClusterOfU*.
   If *AttackType* is empty then *AttackType* = unknown.
   Add *SomeClusterOfU* to $C_{AttackType}$.
4. Merging of the clusters.
   For each cluster in *C*:
   Merge all of the clusters in *U* correspond to the $C_{AttackType}$ set.

FIGURE 5: The cluster merging algorithm.

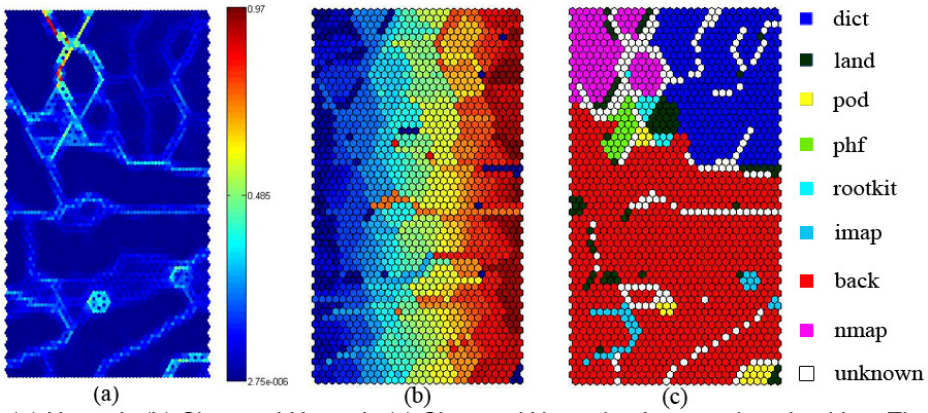

FIGURE 6: (a) U-matrix (b) Clustered U-matrix (c) Clustered U-matrix after merging algorithm. The number of clusters in (b) was 253 and after merging algorithm in (c) reduced to 9 clusters.

- **Classifyin**
- **g of Test Data**

Test data is given to SOM in this unit. It is expected that all the given data vectors from alerts with attack label are placed in the corresponding clusters. Data vectors from false positive alerts should be placed in unknown cluster, otherwise network error should be high. This error shows the distance between entered data vectors and found BMUs. If the error is more than threshold α (α is a constant value) the corresponding data vector is supposed to be as a false positive alert and classified as an unknown cluster.

4. EXPERIMENTAL RESULTS

Matlab software is used to implement the system and SOM toolbox is used to simulate SOM. The map size is 30*60 with grid topology and neighborhood is hexagon. Training data contains 6053 data vectors or 70% of total filtered alert data vectors. In training phase only the data vectors of true positive alerts are used. Since the SOM should be trained with the data vectors extracted from true positive alerts, the data vectors of the false positive alerts are ignored in the training phase. Test data includes 30% of the data vectors of labeled alerts; it means 2591 data vectors, and 2591 data vectors of false positive alerts. The reason for adding the false positive alerts to the test dataset is that always IDSs produce this type of alerts alongside true positive alerts. The number of clusters is 9. Here we let α= 0.1 as a threshold value.

To evaluate the performance of the system, 8 criteria were used. (Table 1)

1-Classification Error (ClaE) is the number of alerts that are wrongly classified. 2-Classification Error Rate (ClaER) is the percentage of wrongly classified alerts (8). 3-Classification Accuracy Rate (ClaAR) is percentage of alerts that are accurately classified as they should be (9). 4-Clustering Error (CluE) is the number of alerts from training data that are wrongly clustered. 5-Clustering Error Rate (CluER) is the percentage of wrongly clustered alerts from train data (10). 6-False Positive Reduction Rate (FPRR) is percentage of false positive alerts that accurately identified and reduced (11). 7-Average Network Hit Error (ANHE) is the average of BUMs error in SOM for all test data (12). 8-Average Network Hit Error for True Positive (ANHETP) is the average of BUMs error in SOM only with the true positive alerts (13).

ClaER=(ClaE ÷Total Number of Alerts Observed)×100 (8)
ClaAR=100–ClaER (9)
ClaER=(CluE÷Total Number of Alerts Observed From Train Data)×100 (10)
FPRR=100–(The Number of False Positive Alerts that Accurately Identified ÷ Total Number of False Positive Alerts Observed)×100 (11)
ANHE=Sum of All BMUs Error÷Total Number of Alerts Observed (12)

ANHETP=Sum of BMUs Error for All True Positive Alerts÷Total Number of Alerts Observed (13)

	ClaE	ClaER	ClaAR	CluE	CluER	FPRR	ANHE	ANHETP
UR	672	12.97	87.03	23	0.38	87.34	0.073	0.011
IUR	33	0.64	99.36	23	0.38	99.71	0.091	0.003

TABLE 1: Proposed system performance metrics.

	ClaE	FPRR	ClaAR
UR	2109	59.70	59.30
IUR	2516	51.87	51.04

TABLE 2: Proposed system performance metrics without considering α threshold value.

As Table 1 shows, the best result is obtained by IUR scaling method which has 99.36% accuracy and its false positive alerts reduction ratio is 99.71%. Because of the IUR method distributes values of attributes of alert vectors as a uniform manner in a mentioned range, the acquired results are better than other unit range method. When Table 1 and Table 2 are compared, you can see the FPRR metric is improved after using α threshold in Table 2. If the result of classification without using α threshold is accepted then gather may exist some incorrectly classified false positive alert vectors. If α threshold is used as a maximum value of network error and classified results have upper value of this threshold value then classification operation is corrected for these alert vectors. Because of dependency between FPRR and ClaAR by improving the result of FPRR metric, the ClaAR metric result is improved.

5. CONSLUSION & FUTURE WORK

In this paper a SOM based system is presented which is able to cluster and classify the alerts with high accuracy. This system is also able to reduce number of false positive alerts considerably.

If the SOM is trained by alerts for various types of attacks with proper filtering process and preprocessing on the alerts, SOM becomes a suitable tool to classify alerts and reduce the false positive alerts in the alert management systems for IDS.
In using SOM for a wide range of alerts with various types of attacks, ratio of error may be high thus using hierarchal SOMs on each trained SOM of a special type of attack can be a solution of this problem.

SOM can be used to find the correlations between alerts. Thus all of the alert vectors with several common attributes placed on one or several neighbour neurons are supposed to be as the related alerts.

6. REFERENCES

1. H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. COMPUT. NETWORKS, 31(8):805-822, 1999. 60 Conclusion And Future Work 61.

2. K. Julisch, "Clustering intrusion detection alarms to support root cause analysis", ACM Transactions on Information and System Security (TISSEC) , 2003, Volume 6 , Issue 4, Pages: 443 – 471.

3. Maheyzah, M. S., Mohd Aizaini, M., and Siti Zaiton, M. H. (2009), " Intelligent Alert Clustering Model for Network Intrusion Analysis.", Int. Jurnal in Advances Soft Computing and Its Applications (IJASCA), Vol. 1, No. 1, July 2009, ISSN 2074-8523. pp. 33 - 48.

4. F. Cuppens. Managing alerts in a multi-intrusion detection environment. Proceedings of the 17th Annual Computer Security Applications Conference, 32, 2001.

5. E. MIRADOR. Mirador: a cooperative approach of IDS. Poster present au me European Symposium on Research in Computer Security (ESORICS). Toulouse, France, octobre, 2000.

6. Kohonen, T, "Self-Organized Maps", Springer series in information. Science Berlin Heidelberg:1997.

7. Kiziloren, Tevfik, "Network traffic classification with Self-Organized Maps", Computer and information sciences, 2007, page(s): 1-5.

8. Pachghare,V. K., "Intrusion Detection System Using Self Organized Maps", Intelligent Agent & Multi-Agent Systems, 2009, page(s): 1-5.

9. Hayoung Oh, Kijoon Chae, "Real-Time Intrusion Detection System Based on Self-Organized Maps and Feature Correlations", Third International Conference on Convergence and Hybrid Information Technology, IEEE, 2008, vol. 2, Pages.1154-1158.

10. Wang, J., Wang, H., Zhao, G. 2006. A GA-based Solution to an NP-hard Problem of Clustering Security Events. IEEE 2093- 2097.

11. Jianxin Wang, Baojiang Cui, " Clustering IDS Alarms with an IGA-based Approach", ICCCAS 2009, pp586-591.

12. Snort: The open source network intrusion detection system. Available: http://www.snort.org/.

13. MIT Lincoln Lab. (1998). DARPA 1998 Intrusion Detection Evaluation Datasets. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html

14. Snort Manual, www.snort.org/assets/82/snort_manual.pdf

15. Matlab Software, http://www.mathworks.com.

16. A. Ultsch, H. P. Siemon, " Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis", Proceedings of International Neural Networks Conference (INNC) (1990), pp. 305-308.

17. Binh Viet Nguyen, "Self-Organizing Map for anomaly detection", Available in http://www.cs.umd.edu/~bnguyen/papers/papers.html

18. SOM Toolbox for Matlab, Available in http://www.cis.hut.fi/projects/somtoolbox/.

19. Juha Vesanto, John Himberg, Esa Alhoniemi, and Juha Parhankangas, "SOM Toolbox for Matlab 5", SOM Toolbox Team, Helsinki University of Technology, 2000.

20. Juha Vesanto and Esa Alhoniemi. Clustering of the Self-Organizing Map. IEEE Transactions on Neural Networks, 11(2):586–600, March 2000.

21. Julisch, K.: Mining alarm clusters to improve alarm handling efficiency. In: Proceeding of the 17th Annual Computer Security Applications Conference, New Orleans, pp. 12–21 (2001)

22. S Terry Brugger and Jedidiah Chow, " An Assessment of the DARPA IDS Evaluation Dataset Using Snort", UC Davis Technical Report CSE-2007-1, Davis, CA, 6 January 2007