What Defines an Intruder? An Intelligent Approach

Hector M Lugo-Cordero, and Ratan K Guha
Dept. of Electrical Engineering and Computer Science
University of Central Florida
Orlando, Florida
Email: {hlugo, guha}@eecs.ucf.edu

***Abstract*—All attacks in a computer network begin with an intruder's action of affecting the services provided to legitimate users. Hence, intrusion detection is vital for preserving integrity, confidentiality, and availability in a computer network. Intrusion detection faces many challenges, such as the need for large amount of data to discriminate between intruders and non-intruders, and the overlapping of user behavior to that of the intruders. This paper aims to target both of these challenges, by employing a distributed intrusion prevention system based on the Binary Partitle Swarm Optimization (BPSO) and Probabilistic Neural Network (PNN) algorithms. Such a system is capable of: 1) locally classifying actions as intruder or non-intruder type, and 2) consulting neighbors for casting a majority vote, upon finding high ambiguity on a decision. The algorithm uses an evolutionary computation approach to select the best features that can help classify intruders, while using fewer amounts of data. Furthermore, the approach uses concepts from semi-supervised learning to improve and adapt over time, to any network infrastructure. To demonstrate the viability of the proposed approach, a random set of data has been selected from the KDD-99 dataset. Such a set contained capture data from both users and attackers. Results have been compared with traditional data mining algorithms from previous work, demonstrating that such a system can have high accuracy, while maintaining a low false alarm rate.**

***Index Terms*—Intrusion Detection; Intelligent Networks**

I. Introduction

Over the last decade, cyber security has become of great importance, since attacks are getting more sophisticated and easy to execute. Through time, it has alarmingly become easier for attackers to steal information at the comfort of their homes, without putting their lives at risk. Hence, intruders in a network must be identified and counter actions must be taken. Failing to do so may result in negative consequences, including identity theft, stealing of credit card numbers, and unavailability of services, among others.

Therefore as a counter measure, intrusion detection systems (IDS) have been proposed to identify malicious users. Moreover, intrusion prevention systems (IPS) take such detection one step further, by taking counter measures, which may block the malicious users from performing their attack. However, intrusion detection/prevention systems (IDPS) face two main challenges: 1) requirement of large amount of data and processing in order to classify behaviors correctly, and 2) the overlapping of actions on masked intruders' behavior. It is because of such challenges, that an IDPS system may incorrectly classify users (or intruders), thus obtaining an undesired outcome.

The problem of Intrusion Detection has been widely studied. Probably the most popular approaches involve data mining tools such as Support Vector Machines (SVMs) [1]. Another recently studied approach is the use of evolutionary computation to evolve optimal classifiers; [2] works with evolving a Fuzzy Logic classifier that can increase the accuracy on dificult classifications. Although some of the previous approaches manage to obtain good results and feature space reduction [3]. Our main interest is to identify intruders correctly by cooperation among neighboring network nodes, and reducing the required dataset by each network node.

This paper presents a distributed cooperative algorithm to detect intruders using a small random set of local data for training purposes. Using a Binary Particle Swarm Optimization (BPSO) algorithm, the best features for classification are selected by each network node locally. Based on such features, a user is probabilistically identified as a normal user or an intruder. Whenever such probabilistic measures are not very definitive, the result is classified as ambiguous. This ambiguity is resolved by consulting neighboring nodes through a voting mechanism.

Classification in the system is handled by each node using a Probabilistic Neural Network (PNN). The PNN compares entries against the distribution of what is believed to be an intuder and what is not. The class which provides a higher probability confidence becomes the classification for the specific node. The implementation of such network has been enhanced for improved adaptation from one node into the other. The mixture of BPSO with PNN make the system more tolerable to noise in the data, and results show how the system is able to correctly classify in most cases, with almost no false alarms.

The rest of the paper is organized as follows. Section II describes the characteristics and challenges of intrusion detection. Following, section III presents alternatives to selecting the best set of features for classification, as well as examples of common features in intrusion detection. Later in section IV, the proposed system is described and its advantages are also presented. Section V presents the experimental setup, along

978-1-4673-5867-5/13/$31.00

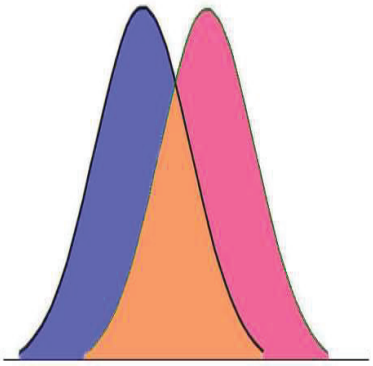

Fig. 1. **Overlapping of User Behaviors**. An IDPS biggest challenge is dealing with false positive (a legitimate user classified as malicious) and false negatives (a malicious user classified as legitimate). Such problem occurs because of outliers which create an overlapping region between the behavior of both types of users.

with the results and discussion in section VI. Finally, previous work that motivated this research is presented in section VII, and section VIII concludes the paper.

II. Intrusion Prevention Systems

Intrusion prevention systems have capabilities to detect an intruder, and block such intruder from accessing the system where it was detected. Detection of the intruders may be done using two primary techniques: signature and statistical based approaches. Signature based intrusion detection rely on common attack signatures, which are stored inside a database, and are consulted to be compared against incoming packets. Such intrusion detection requires noiseless data and continuous signature updates to achieve improved detection accuracy.

Meanwhile, statistical based detection employs analysis of typical user behavior and attempts to find anomalies in traffic; such anomalies may correspond to intruders in the network. Problems with such approach arise from the fact that typically there is a lot of data of valid users, while at the same time, little is known about intruders' data. Furthermore, as Figure 1 shows, there might be areas where legitimate users' behavior overlaps with intruders' behavior.

The next section describes common characteristics that may serve to classify between an intruder and non-intruder.

III. Feature Extraction in IDPS

A good feature extraction mechanism is essential in all classification problems. Extracting features reduces the dimensionality of data, thus, providing an easier classification. The better the quality of the features extracted, the better the accuracy of the classification system is obtained; this is because a clearer separation between classes is achieved. Different strategies exist, for gathering the best features: empirical [4], Principal Component Analysis [5], and Evolutionary Computation algorithms [6], [7]. Empirical analysis is easy to perform, but requires multiple tests, which have to be analyzed by an expert that can decide which are the best features to be used, based on current data. The problem with employing current data is that the data might change in the future. Principal Component Analysis (PCA) automates such process by using a set of equations that can help discriminate features, and select the ones that provide the most information; however, PCA is highly affected by noise in the data [7].

The last strategy uses an evolutionary algorithm (EA), such as the Binary Particle Swarm Optimization, to search for the best feature set. EAs have the characteristic of being less affected by noise, as well as being implicitly parallel, allowing them to search multiple possible solutions at the same time. This work benefits from this nature, in order to run a distributed BPSO algorithm across all intrusion detection nodes in the network. Typical features for intrusion detection include:

- # of IP Addresses in the network
- Avg. interval between packets
- # of protocols in the network
- # of protocols used by a single host
- Type of protocol requested
- # of packets with 0 length size
- Avg. data length
- # of non-ASCII characters in data
- Time interval of packets
- Avg. error rate

The next section discusses the proposed methodology and its advantages.

IV. Methodology

This section describes the methodology proposed for an improved classification of intruders and non-intruders, taking advantage of the network capabilities to communicate among neighbors. The system is composed of three components (Figure 2): features discrimination, behavior classification based on selected features, and taking action whenever an intruder is detected.

A. Feature Selection

Choosing the best features, given a large amount of data, can be a challenging task. Instead of obtaining empirical results, the proposed system employs a Binary Particle Swarm Optimization (BPSO) approach. The BPSO has been utilized in the past for determining the best features to be used for classification problems [6]. In previous work, the BPSO has been proven to be less affected by noise; therefore, the BPSO can find features which better represent the data. The goal of the algorithm is to maximize the inter-class variance given by:

$$F = \sqrt{\sum_{i=1}^{L} (M_i - M_o)^t (M_i - M_o)}$$

where M_i represents the mean for each class of features, and M_o the overall mean, considering a number of L distinct classes (in this work 2, intruder or not).

Moreover, the BPSO finds a binary mask T, which can be applied as a point-by-point product between the bits in the mask and the elements of the feature vector. To find

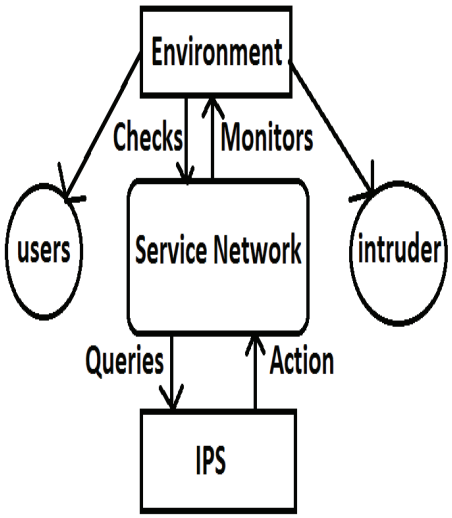

Fig. 2. **Adaptive IDPS Infrastructure**. The service network scans the environment, where there may be users and/or intruders. To correctly detect whos an intruder, the system consults with the adaptive IDPS, which replies with an action to take. Such action may be to do nothing, if the user is legitimate.

such mask, the BPSO starts with a collection of randomly initialized masks, known as particles. These particles have a velocity associated with each one, which determine how fast they explore the sample space on each generation. During each generation and for each particle, the particle velocity is updated according to some inertia factor α, which prevents drastic changes in the velocity. Particles also consider their past individual knowledge and the knowledge gained by other particles. The importance that each particle gives to such knowledge can be set with some influence constants c_1 and c_2 respectively. Once the velocities are updated, each particle will set a bit as 1 (select the corresponding feature) or 0 (neglect the corresponding feature), following a sigmoid function of the particle velocity. To ensure that the solutions become stable, the velocities should be kept between a finite interval $[-d, d]$. After some empirical trials we have found that $d = 4$ is an adequate value for such interval. Otherwise, values smaller than -4 would prevent a bit from being set to 1, while values larger than 4 would prevent bits from being set to 0. Finally, the updated particles are compared in fitness (inter-class variance) and replace old individual/global best if their fitness is higher. The complete detailed procedure can be seen in algorithm 1.

In the proposed system, each node in the network runs a local BPSO, with its own local data. Thus, the amount of data required is less, and nodes can communicate with neighbors whenever high uncertainty about a decision is present. Such a process will be better described in the next sub-section, when the Probabilistic Neural Network classifier is discussed.

Algorithm 1 The Binary Particle Swarm Optimization

```
Init population of size P randomly
Init particles velocities v to 0
Init individual best to current population
Init global best to argmax_particle{F(particle)}
Set individual influence constant c1 (e.g., 2)
Set global influence constant c2 (e.g., 2)
Set α inertia factor to a desired step size (e.g., 0.9)
while Generations Remain AND Value not reached do
    for Each Particle P do
        v ← α · v + c1 · rand1 · (individual_best − particle) +
        c2 · rand2 · (global_best − particle)
        if v > 4 then
            v ← 4
        end if
        if v < −4 then
            v ← −4
        end if
        for Each bit do
            if rand < 1 / (1 + e^−v) then
                bit ← 1
            else
                bit ← 0
            end if
        end for
        Evaluate particle
        if F(particle) > F(individual_best) then
            individual_best ← particle
            if F(particle) > F(global_best) then
                global_best ← particle
            end if
        end if
    end for
end while
```

B. Behavior Classification

Classifying the behavior of users (or intruders) requires some confidence or probability, that tells the node how much is known about a packet. The Probabilistic Neural Network (PNN) becomes a good classifier, using the best features selected by the BPSO, which filters out some or most of the possible noise in the data.

PNNs create statistical data, mean μ and standard deviation σ, for each sample features. Such statistical data is used to compute the euclidean distance between input feature vector and the sample feature vector. The distance value is used as argument for a radial basis function that describes how related is the input vector to the samples (i.e., intuder/non-intruder). The total probability for each class (i.e., intruder/non-intruder) is computed at the end and the larger of these two values determines the classification.

PNNs can prune unneeded comparison samples, making the classification process faster. Since the PNN uses a statistical approach to classify its subjects, it is easy to update the mean μ and standard deviation σ of each feature, such that the amount of times revisiting the data is reduced, as follows:

$$\mu_{new} = \frac{N_{old} \cdot \mu_{old} + newSample}{N_{new}}$$

$$\sigma_{new} = \sqrt{\frac{\sum_i (x_i - \mu_{new})^2}{N_{new}}}$$

where N_{old} is the number of samples before the new observed feature vector value $newSample$ is considered, N_{new} is the amount of samples in the new dataset (i.e., $N_{new} = N_{old} + 1$).

Upon encountering high ambiguity in the classification (probabilities close to 0.5), nodes follow the procedure shown in figure 3, to find a better classification, based on the results from neighbors classifications. Once the neighbors cast their votes, the classifying node counts the number of votes per class. The class that gets the most number of votes is the resulting classification. Such decision involves a learning phase in the analyzing node, to adapt to the previously ambiguous packet.

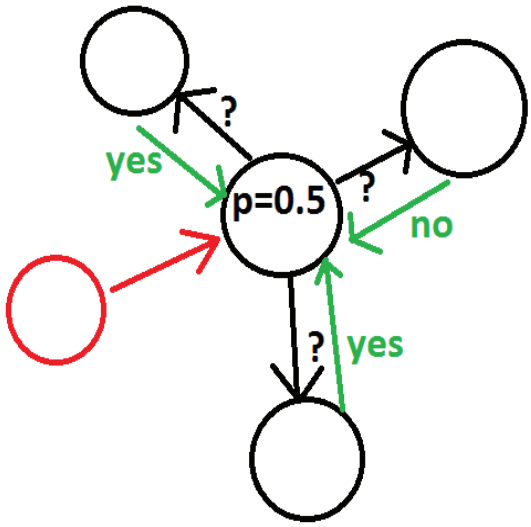

Fig. 3. **IDPS Disambiguation**. Upon encountering a classification with high ambiguity, the system proceeds to consult its neighbors, which reply back with their vote; such vote is represented by the probability of the entity being an intruder. Once their reply is received, the system adapts to the situation by creating a new sample, based on the features of the ambiguity case, and the result of the majority vote as target classification.

C. Types of Actions

Once an intrusion is detected, IDPS can take various actions to protect the network and its users' from malicious attacks. Such actions are:

- Log: record the intrusion event, along with all data that can identify the intruder
- Alert: alert others (administrator and neighbors) by sending email and/or packet to other nodes in the network, such that they may limit or block access of the intruder
- Limit: limit the bandwidth of the intruder, in case there is the possibility that is a legitimate user, who is incorrectly utilizing the network
- Block: block all connections from the intruder

It is important to notice that more than one of such actions can be applied to the same event, depending on the severity of the event. The next section discusses the experimental setup to test the performance of the proposed IDPS.

V. Experimental Setup

This section describes the experimental setup for testing the performance of the proposed IDPS. The BPSO and PNN algorithms were implemented in Java. For training and testing data, the dataset from the third International Knowledge Discovery and Data Mining Tools Competition [8] was used.

Such data has been used for IDS benchmarks, dealing with data mining approaches [1], feature extraction [3], and evolutionary computation [2]. The data contains both normal users and common network attacks in a simulated military environment. Among such attacks are: rootkits, smurf, buffer overflow, and others. The features selected were limited to those with continuous values, for a total of 34 features. Features with symbolic values, such as the service and protocol type, were discarded. Nodes were trained by dividing at random the amount of the total data among them (i.e., simulated nodes). Around one fifth (20%) of the assigned was used for testing purposes.

Two experiments were used for comparison. In the first experiment, the data was processed using by only node using the BPSO feature selection and PNN classifier. In the second experiment, the training was done locally by 5 IDPS, using distributed local data. In this experiment, during testing, the same node from the first experiment consulted the additional IDPS nodes when ambiguous cases were found. During such cases, the additional IDPS nodes casted their classification vote, and the classification that earned the most votes won.

The results obtained and presented in the next section, were studied at the central classifier (i.e., central node), using average performance of 10 trained classifiers (i.e., 10 runs). Neighbors (the other 4 nodes) were only consulted for classification upon finding ambiguity. The experiment is done with 5 running processes, so no network communication overhead is studied at this point. All nodes were trained with 1000 random entries from the dataset, labeled both as normal or some type of attack (e.g., rootkit, spy, nmap, smurf, buffer overflow, ipsweep, etc.). Different types attacks, were considered as an intruder regardless of the attack type. For testing, a random number of testing samples was generated, centered each run at 200 samples. The features contained in the samples included: source bytes, destination bytes, number of failed logins, number of created and access files, error rates, among others.

Next section discusses the results obtained by the system.

VI. Results and Discussion

In this section, we first discribe the results without the majority vote.

Table I presents the confusion matrix for the no majority casting vote. The Ambiguous column states the percent of ambiguous cases, where a user acted as an intruder (2%), or an

intruder as acted as user (40%). Out of such ambiguous cases, the system sometimes made a correct classification, sometimes not. Such impact may be seen in the User and Intruder columns, which give the total rate per scenario. Although the accuracy was not too high, it was necessary to try out different set of features from the beginning. Hence, it was vital to observe what features the BPSO filter selected, which allowed correct classification of the ambiguous cases. Furthermore, the system had no problem identifying the normal users, where almost a perfect score was obtained. The challenge arose with the intrusion detection scores, which included a large amount of ambiguous classifications that resulted in a relatively low accuracy.

TABLE I
CONFUSION MATRIX WITHOUT MAJORITY VOTES

Actual/Classification	User	Intruder	Ambiguous
User	0.97	0.03	0.02
Intruder	0.18	0.82	0.4

Detection Rate 0.82 **False Alarm Rate** 0.03

In the second experiment, majority vote was added for improving the ambiguous classifications. The results of adding the majority vote to the same IDPS are presented in Table II. The ambiguous cases are not shown because the two runs were seeded equally. Such configuration makes the ambiguous cases be the same as the run without majority vote. However, with majority vote, the accuracy of the IDPS was increased. It is to be noted that an odd number of neighboring nodes avoids a tied result.

TABLE II
CONFUSION MATRIX WITH MAJORITY VOTES

Actual/Classification	User	Intruder
User	0.99	0.01
Intruder	0.02	0.98

Detection Rate 0.98 **False Alarm Rate** 0.01

We have also traced the correct classification rate in both experiments. This trace value is shown in Figure 4. The blue and red traces are for the first and second experiment respectively. A point to noted that first traces were all normal users, having correct classification. The trace also shows the improvement in the classification rate in the second experiment.

In order to validate our experiment, we have examined the results presented in [1]. For comparison, we have presented the results of [1] with out results in Figure 5. In [1], Support Vector Machine (SVM) provided poor performance. On the other hand, Naïve Bayes, Random Tree, Random Forest, and Multilayer Perceptron showed a 82%, 93%, 93%, and 92% respectively. Our first experiment's results compares evently with Naïve Bayes result, but the second experiment shows significant improvement in the classification compared to the other methods. However, more detailed work needs to be studied with overhead for network neighbors cooperation.

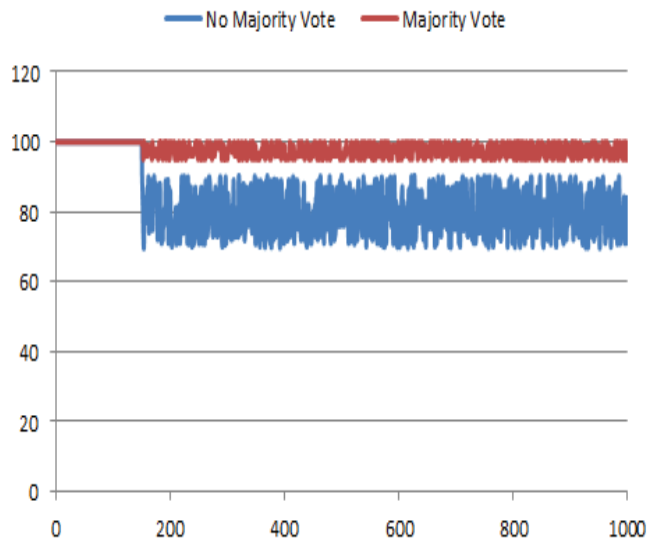

Fig. 4. **IDPS Detection Rate**. Majority votes helped to correct cases where the difference between an intruder and a normal user were ambiguous. Initially both systems had the same level of accuracy because no attack was present at the beginning of data.

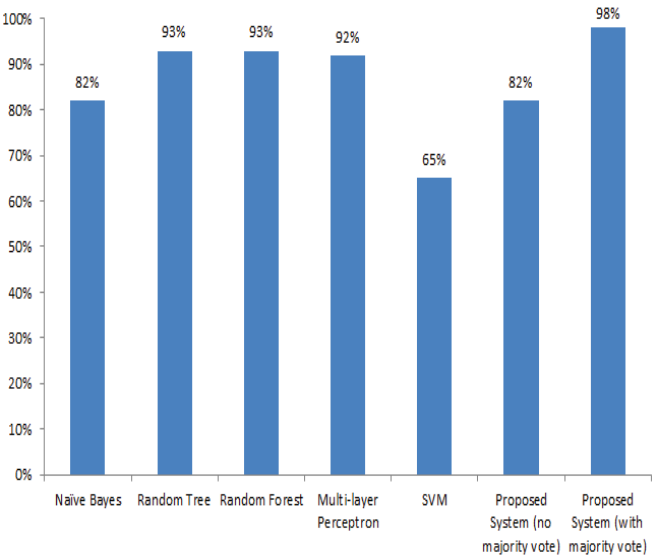

Fig. 5. **IDPS Comparison**. Good accuracy is possible using common machine learning tools. Without majority vote, the system may not perform as well as it can; majority vote helps boost ambiguous classification results.

The next section presents some related work that help guide this research.

VII. RELATED WORK

Intrusion detection has been a topic of interest for many researchers of different areas. The feature extraction to classify transmissions can be found among the most common interests. Techniques like Fuzzy Logic [9], [4], Principal Component Analysis [10], Generalized Discriminant Analysis [5], and Genetic Algorithms [11], [12], have been satisfactorily used.

The BPSO surpasses traditional GAs because its convergence speed is faster, and generates more feasible solutions, by having multiple mechanisms to avoid local optima [7], [6].

Different characteristics of IDPS were studied in [13], among them the differences between host-based [14] vs network-based [15] IDPS. The approach used in this work is both host-based because it runs locally (although scanning the network traffic), and network-based when the IDPS requires the majority votes from its neighbors.

Other approaches involve analyzing behavior through signature detection [16], [17], [18]. Our research will consider as future work what benefits majority vote can acquire via such approaches.

The next section concludes the paper.

VIII. Conclusion

Intrusion prevention can help preserve the integrity, availability, and confidentiality of data in a network. Computer networks can benefit from their distributed nature, by distributing the data samples, and only querying for more data when needed; hence, solving the intrusion detection with less amount of data. The proposed cooperative system employed a BSPO algorithm as feature selector, along with a PNN classifier. Both being robust against noise data, the classification of behavior became more consistent in accuracy. The cooperation among neighboring nodes in the system was able to reduce the number of false alarms, without sacrificing accuracy in intrusion detection.

References

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, july 2009, pp. 1 –6.

[2] J. Zhong, H. Wu, and Y. Lai, "Intrusion detection using evolving fuzzy classifiers," in *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*, vol. 1, aug. 2011, pp. 119 –122.

[3] V. Boln-Canedo, N. Snchez-Maroo, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5947 – 5957, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410012650

[4] O. Linda, M. Manic, T. Vollmer, and J. Wright, "Fuzzy logic based anomaly detection for embedded network security cyber sensor," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 202 –209.

[5] S. Signh, S. Silakari, and R. Patel, "An efficient feature reduction technique for intrusion detection system," in *Machine Learning and Computing, 2009. International Conference on*.

[6] H. Lugo-Cordero, A. Fuentes-Rivera, R. Guha, K. Lu, and D. Rodriguez, "Multimodal species identification in wireless sensor networks," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on*, dec. 2011, pp. 385 –388.

[7] M. Tariquzzaman, J. Y. Kim, and S. Y. Na, "Pso based optimized reliability for robust multimodal speaker identification," in *Proceedings of the 4th WSEAS international conference on Circuits, systems, signal and telecommunications*, ser. CISST'10. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 157–162. [Online]. Available: http://portal.acm.org/citation.cfm?id=1807993.1808025

[8] "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html," *KDD Cup 1999 Data for The Fifth International Conference on Knowledge Discovery and Data Mining*.

[9] J. Xin, J. Dickerson, and J. Dickerson, "Fuzzy feature extraction and visualization for intrusion detection," in *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, vol. 2, may 2003, pp. 1249 – 1254 vol.2.

[10] G. Kuchimanchi, V. Phoha, K. Balagani, and S. Gaddam, "Dimension reduction using feature extraction methods for real-time misuse detection systems," in *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, june 2004, pp. 195 – 202.

[11] C. Bacquet, A. Zincir-Heywood, and M. Heywood, "Genetic optimization and hierarchical clustering applied to encrypted traffic identification," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 194 –201.

[12] T. Vollmer, J. Alves-Foss, and M. Manic, "Autonomous rule creation for intrusion detection," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 1 –8.

[13] F. Sabahi and A. Movaghar, "Intrusion detection: A survey," in *Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on*, oct. 2008, pp. 23 –26.

[14] F. Barbhuiya, S. Biswas, N. Hubballi, and S. Nandi, "A host based DES approach for detecting arp spoofing," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 114 –121.

[15] S. Yu and D. Dasgupta, "An effective network-based intrusion detection using conserved self pattern recognition algorithm augmented with near-deterministic detector generation," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 17 –24.

[16] E. Ferragut, D. Darmon, C. Shue, and S. Kelley, "Automatic construction of anomaly detectors from graphical models," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 9 –16.

[17] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 245–254. [Online]. Available: http://doi.acm.org/10.1145/586110.586144

[18] Y. Hu, C. Frank, J. Walden, E. Crawford, and D. Kasturiratna, "Profiling file repository access patterns for identifying data exfiltration activities," in *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, april 2011, pp. 122 –128.