Hybrid Fuzzy Adaptive Wiener Filtering with Optimization for Intrusion Detection

Revathi Sujendran and Malathi Arunachalam

Intrusion detection plays a key role in detecting attacks over networks, and due to the increasing usage of Internet services, several security threats arise. Though an intrusion detection system (IDS) detects attacks efficiently, it also generates a large number of false alerts, which makes it difficult for a system administrator to identify attacks. This paper proposes automatic fuzzy rule generation combined with a Wiener filter to identify attacks. Further, to optimize the results, simplified swarm optimization is used. After training a large dataset, various fuzzy rules are generated automatically for testing, and a Wiener filter is used to filter out attacks that act as noisy data, which improves the accuracy of the detection. By combining automatic fuzzy rule generation with a Wiener filter, an IDS can handle intrusion detection more efficiently. Experimental results, which are based on collected live network data, are discussed and show that the proposed method provides a competitively high detection rate and a reduced false alarm rate in comparison with other existing machine learning techniques.

Keywords: Fuzzy rule generation, Wiener filter, simplified swarm optimization, intrusion detection, fuzzy membership function.

Manuscript received Mar. 4, 2014; revised Jan. 24, 2015; accepted Feb. 3, 2015.

Revathi Sujendran (revathisujendran86@gmail.com) is with the Department of Computer Science, Government Arts College, Bharathidasan University, Tamilnadu, India.

Malathi Arunachalam (corresponding author, malathi.arunachalam@yahoo.com) is with the Department of Computer Science, Government Arts College, Coimbatore, India.

I. Introduction

Internet usage has become increasingly widespread in relation to online banking, shopping, stock exchanges, and online auctions. Due to its open public nature, the security of the Internet and any related data is always an issue. The tremendous growth of Internet use has led to the development of network intrusion detection, which is especially critical for protection. Intrusion detection is mainly used to identify malicious activity within a network or system; such activity makes network resources vulnerable to attacks [1].

Intrusion is mainly based on either misuse detection or anomaly detection. Regarding misuse detection, it searches for a pattern of user behavior that matches two well-known scenarios. Meanwhile, anomaly detection is based on the normal behavior of network data, and it detects attacks based on significant deviations from the normal data flow [2]. The main advantage of anomaly detection is that it detects attacks when there is no existing signature. However, comparatively, it suffers from a larger false positive rate.

Moreover, it is very difficult to train a computer system using an intrusion detection method, such as fuzzy rule generation combined with a Wiener filter, using only unclassified data as, ultimately, this will lead to a drop in accuracy.

Recently, many *soft* computing methods, such as fuzzy sets, neural networks [3], genetic algorithms (GAs) [4], and simulated annealing, have been used in the field of intrusion detection [5]–[6]. Broadly speaking, the fuzzy concept is one where the knowledge of human experts is used to create a fuzzy rule; such a fuzzy rule will then make an intrusion detection system (IDS) more robust [7]; however, such an IDS will still be deficient with regards to learning and training adaptation. It is this fact that has led to the development of

http://dx.doi.org/10.4218/etrij.15.0114.0275

automatic fuzzy rule generation, such as neuro-fuzzy rule generation and genetic-fuzzy rule generation.

From the point of view of classification, an intrusion detection module is mainly used to identify data as either normal or abnormal (that is, an attack). In addition, machine learning techniques, such as data mining [8]–[9], statistical analysis [10], immunological-inspired techniques [11], and computational intelligence algorithms, such as genetic programming [12], artificial immune systems [11], and swarm intelligence (SI), are also reported in [6]–[13] as a means for solving the IDS problem. Among the aforementioned methods, particle swarm optimization (PSO) is one of the most popular SI algorithms. Over a series of tasks, PSO is better at optimization than GAs, particularly in optimization applications [14]; however, very little research has been performed on applying the PSO technique to solving network intrusion detection problems. As a result, this paper focuses on a new simplified version of PSO (namely, simplified swarm optimization (SSO)) along with automatic fuzzy rule generation for detecting intrusion patterns.

In this paper, a live network dataset is used both for training and for testing attacks. Such network data are collected using Linkware Technologies as a live dataset [15]. This is a leading provider of embedded systems and is used as a network simulation capture (NSC) simulation tool to capture live data for analyzing various attacks in a network. This research paper has various levels to detect intrusion. First, it selects the most relevant attribute that represents a pattern of network intrusion using a new hybrid SSO with random forest algorithm (SSO-RF). Second, an automatic fuzzy rule is generated based on a Wiener filter as a decision-making process to detect and classify current intrusion activity as normal or as an attack. Finally, SSO is employed to optimize the structure of the fuzzy sets of the fuzzy decision-making engine. The structure of the proposed work is summarized as follows:

1. Fuzzy rule generation based on a Wiener filter is used to collect and analyze network data to detect anomalous behavior in network traffic.
2. The hybrid feature-selection method (SSO-RF) is used to reduce the attributes more efficiently.
3. The proposed adaptive fuzzy granule fitness function is used to mine more new rules with higher accuracy.
4. The proposed framework is more flexible in detecting intrusion behavior in real-time scenarios.
5. A high detection rate (DR) and low false alarm rate (FAR) are obtained for the overall network data.

The rest of this paper is organized as follows. Section II describes some of the related work based on various intrusion detection methods. Section III explains the concept of fuzziness and the Wiener filter in relation to detection. Section IV discusses the proposed system architecture. Section V draws some experimental results for analysis. Finally, the conclusions and future work are detailed in Section VI.

II. Related Work

IDS research has been promoted based on datasets developed by the Defense Advanced Research Project Agency (DARPA) and Air Force Research Laboratory (AFRL) of the Lincoln laboratory at MIT [16], which are widely used as training and testing datasets for evaluating IDS. Later, the Knowledge Discovery database developed, as a standard benchmark, the KDDCup99 dataset, which is commonly used to detect intrusion [17]. It consists of five million single connection records of training data and two million connections for testing data. Due to its huge size, only 10% of this dataset is used for IDS. However, various statistical issues [18] degrade the performance of KDDCup99 — a fact that led to the creation of the NSL-KDD dataset [19], which consists of only selected records from the KDDCup99 data, whereby no redundant or duplicate records are present in either the training data or the testing data.

A.N. Toosi and M. Kahani [20] have proposed a neuro-fuzzy classifier based on an adaptive neuro-fuzzy inference system using GAs. In addition, subtractive clustering is used to group data from large databases. No feature selection methods were used. The training time, testing time, and cost per example of the method proposed by A.N. Toosi and M. Kahani exhibited better accuracy than any other existing system.

Y.Y. Chung and N. Wahid [21] have proposed a hybrid network IDS using SSO. The proposed system achieves higher classification accuracy than others, with 93.3%, and it is a competitive classifier for IDS.

S. Mabu and others [22] have proposed an ID model based on fuzzy class association (FCA) rule-mining using genetic network programming (GNP). Experimental results based on FCA and GNP and using the KDDCup99 and DARPA98 databases show that it affords competitively high DRs compared with other learning techniques.

S. Zaman and others [23] have proposed a new concept for feature selection based on various soft computing methodologies, such as PSO, GA, and differential evolution (DE), for reducing attributes in the KDD dataset to detect intrusion. The classification accuracy, FAR, training time, and testing time are classified using SVMs and NNs, and it was found that DE reduces more attributes than the other two approaches.

M. Davarynejad and others [24] proposed a new adaptive fuzzy granule fitness function to generate a fitness queue for various evolutionary optimization techniques, such as GAs,

producing better results.

A.J. Malik and others [25] have proposed a new concept — hybrid PSO with a random forest algorithm — to reduce the number of required feature attributes. Their new concept achieves an accuracy of approximately 95%.

M. Al-Kasassbeh [26] proposed a new concept for a Wiener filter as an agent to filter out various attacks that pass through a network. The main use of this filter is to eliminate the scalability issues based on a statistical approach.

III. Fuzziness and Wiener Filter

1. Fuzzy System

Fuzzy systems are rule-based systems that can be easily structured based on prior knowledge. A fuzzy system is based on fuzzy logic, which provides a computational framework for manipulating and reasoning about imprecise expressions of knowledge. L.A. Zadeh [27] explains that fuzzy logic is an extension of Boolean logic in that either a full-member set or a non-member (*crisp*) set can be used for making complex decisions. The elements of a fuzzy set are based on a triangular membership function, $\mu(x)$, with a value range between [0, 1].

$$\mu_A(x) = \begin{cases} 1 & x \in A, \\ 0 & x \notin A. \end{cases} \quad (1)$$

A fuzzy rule is generally based on a fuzzy If-Then rule that generates rules on a given input "If" part as a fuzzy antecedent and a "Then" part as a fuzzy consequent.

Seven linguistic values are used for each attribute — very small (VS), small (S), small-medium (SM), medium (M), large-medium (LM), large (L), and very large (VL). Figure 1 shows the membership functions (MFs) of the seven linguistic values.

1. *Fuzzification*. Convert classical data or crisp data into fuzzy data or MFs.
2. *Rule Evaluation*. Each fuzzy rule is determined and based on a degree of membership of crisp input values in the fuzzy set of the antecedent.
3. *Defuzzification*. It transposes the fuzzy outputs into crisp values.

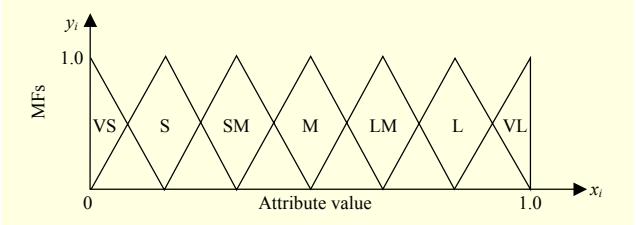

Fig. 1. MFs of seven linguistic values.

2. Weiner Filter

A Wiener filter is used to discriminate an original signal from a noisy signal. It is employed in a wide range of applications, such as echo cancellation, linear prediction, signal restoration, channel equalization, and system identification [28]. The coefficient of a Wiener filter is designed to minimize the mean square error (MSE) between the desired output and the actual output from the filter. A Weiner filter is an optimum linear filter, consisting of an estimation of a desired signal sequence from another related sequence. It is used to calculate statistical parameters, such as the mean and the correlation function, based on the original signal with an unwanted additive. The problem for such a linear filter is the requirement to minimize the effect of noisy data at the output according to a statistical criterion.

In this research work, a Wiener filter is used to filter out and is based on the minimum mean square error (MMSE). Based on fuzzy rule generation, the Wiener filter is trained in relation to error detection and postulates abnormalities that exist within network data.

The Weiner filter input consists of a time series, $x(0)$, $x(1)$, $x(2)$, … , $x(n)$, and the filter is itself characterized by the impulse response time as w_0, w_1, w_2, … ; furthermore, at some discrete time, n, the filter produces an output denoted by $y(n)$. This output is used to provide an estimate of a desired response time denoted by $d(n)$. With the filter input and the desired response time representing single realizations of their respective stochastic processes, the estimation is accompanied by an error with statistical characteristics of its own.

In particular, the estimation error, $e(n)$, is as small as possible in a statistical sense. When the Wiener filter (see Fig. 2) operates under optimum conditions, it takes on the following special form:

$$e_{\min}(n) = d(n) - y_{\text{opt}}(n), \quad (2)$$

whereby the terms may be rearranged as follows:

$$d(n) = y_{\text{opt}}(n) + e_{\min}(n). \quad (3)$$

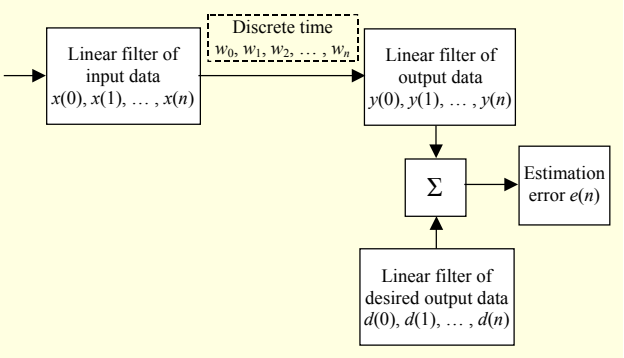

Fig. 2. Block diagram of Wiener filter.

http://dx.doi.org/10.4218/etrij.15.0114.0275

Let J_{MSE} denote MMSE, which can be defined as follows:

$$J_{\mathrm{MSE}} = \mathrm{E}[|\, e_{\min}(n)\,|^2]. \tag{4}$$

Hence, the mean square values of both sides of (3) are evaluated, where $e_{\min}(n)$ is the estimation error. Next, taking the estimation problem of finding the vector $\mathbf{w}$ that optimizes the cost function $J_{\mathrm{MSE}}(\mathbf{w})$ and applying it to the principle of orthogonality, we get the following equation (5), where $Lx1$ represents the optimum filter length:

$$\mathrm{E}[e_{\min}(n)X(n)] = \mathrm{E}\left[d(n) - w_{\mathrm{opt}}^{\mathrm{T}}X(n)\right] = 0_{Lx1}, \tag{5}$$

where we can interpret from the principle of orthogonality that at time n the input vector $\mathbf{x}(n) = [x(n),\ x(n-1)]^{\mathrm{T}}$ will pass through the optimal filter $w_{\mathrm{opt}} = [w_{\mathrm{opt},0},\ w_{\mathrm{opt},1}]^{\mathrm{T}}$ to generate the output $y_{\mathrm{opt}}(n)$. Given $d(n)$, $y_{\mathrm{opt}}(n)$ is the only element in the space spanned by $x(n)$ that leads to an error $e(n)$ that is orthogonal to $x(n)$, $x(n-1)$, and $y_{\mathrm{opt}}(n)$. The computational optimal solution is

$$\mathrm{E}[X(n)X^{\mathrm{T}}(n)]w_{\mathrm{opt}} = \mathrm{E}[X(n)d(n)]. \tag{6}$$

We introduce the following definitions for the input auto-correlation matrix and the cross-correlation vector, respectively. The two expectations in (6) may be interpreted as follows:

$$\begin{aligned} \mathbf{R}_{XX}(k) &= \mathrm{E}[X(n)X^{\mathrm{T}}(n)], \\ \mathbf{r}_{xd}(k) &= \mathrm{E}[x(n)d(n)]. \end{aligned} \tag{7}$$

In the expectation $\mathbf{R}_{XX}(k)$, we have

$$\mathbf{R}_{XX}(k) = \mathrm{E}[X(n)X^{\mathrm{T}}(n)] = \sum_{i=0}^{L-1} \mathrm{E}[x(n-k)x(n-i)] \quad k = 0,\ 1,\ 2,\ \ldots \tag{8}$$

The signal statistic, $\mathbf{R}_{XX}(k)$, is an auto-correlation function of the filter input $x(n)$ for a lag of i-k in (8). For the expectation $\mathbf{r}_{xd}(k)$, we have

$$\mathbf{r}_{xd}(k) = \mathrm{E}[x(n)d(n)] = \mathrm{E}\left(x(n-k)d(n)\right) \quad k = 0, 1, 2, \ldots, \tag{9}$$

where we let $\mathbf{r}_{xd}(k)$ be a cross-correlation vector between the filter input $x(n-k)$ and the desired response $d(n)$ for a lag of $-k$ in (9).

The auto-correlation matrix $\mathbf{R}_{XX}$ is defined as $\mathbf{R}_0$; then the equation for the optimum filter coefficient $\mathbf{w}$ can be written as follows:

$$\mathbf{w}\mathbf{R}_0 = \mathbf{r}_{xd}, \tag{10}$$

$$\mathbf{w}\mathbf{R}_0^{-1} = \mathbf{r}_{xd}, \tag{11}$$

where $\mathbf{r}_{xd}$ is the cross-correlation vector between the input sequence $x(n)$ and desired output sequence $d(n)$. In addition, $\mathbf{r}_{xd}$ is denoted as an L-by-1 cross-correlation vector between the tap inputs $x(n)$, $x(n-1)$, … , $x(n-L+1)$ of the filter and the desired response $d(n)$ as follows:

$$\mathbf{r}_{xd} = \left[p(0), p(-1), \ldots, p(1-L)\right]^{\mathrm{T}}, \tag{12}$$

where p is a joint wide sense stationary process. The correlation matrix can be written as

$$\mathbf{R} = \begin{bmatrix} r(0) & r(-1) & r(-2) & r(-3) \\ r(1) & r(0) & r(-1) & r(-2) \\ r(2) & r(1) & r(0) & r(-1) \\ r(3) & r(2) & r(1) & r(0) \end{bmatrix}, \tag{13}$$

where the correlation matrix $\mathbf{R}$ is a Toeplitz matrix [29]–[30]. The matrix elements along its main diagonal are all equivalent to each other, as are those that are along diagonals that are parallel to the main diagonal. The correlation matrix $\mathbf{R}$ is always positive. To calculate the variance, which is based on (4), MMSE is set equal to the difference between the variance of the desired response and the variance of the estimate that the filter produces as its output; that is,

$$J_{\mathrm{MSE}} = \sigma_d^2 - \sigma_{y_{\mathrm{opt}}}^2. \tag{14}$$

To find the normal and abnormal states of the network, the auto-correlation matrix $\mathbf{R}$ utilizes the eigen-decomposition to identify normal and abnormal states. For instance, an eigenvector marks the boundary between normal data traffic and abnormal traffic based on minimum and maximum values. The values beyond the maximum may lead to an abnormality in the data flow.

IV. Proposed System Architecture

1. Feature Reduction

Live network server data is collected based on a network data recorder (or sniffer). It reads raw network packets upon transmission and stores them onto a disk. This unclassified data is then separated into either attack data, which is based on various attack categories (DOS, Probe, U2R, and R2L), or into normal data. These datasets are then stored in a database, which contains around 20,971,520 records, and are used for both the training and testing of the IDS. There are more than 13,000 attacks in the training and testing dataset that cover the four major attack categories. This dataset contains 41 attributes, some of which may be used to detect as opposed to identify intrusions. Table 1 shows a number of records in the dataset.

The traditional pre-processing algorithms, such as particle swarm optimization with random forest (PSO-RF), DE, genetic algorithm, and so on, are not adaptive to benchmark datasets, such as KDDCup99, DARPA, and NSL-KDD, due to the huge sizes of such datasets; thus, if they were to be applied

Table 1. Number of records in real-time dataset.

Attack name	Number of records
Normal	5,917,854
DOS	7,683,641
U2R	165,920
R2L	3,785,161
Probe	3,418,944
Total	20,971,520

to such datasets, then this would result in large numbers of false recommendations.

In this paper, a new hybrid technique, SSO-RF, is used. SSO is a simplified version of PSO and can be used to find the global minimum of nonlinear functions [21]. This approach is used to solve the problem of classification and reduce the dimensions of a dataset. Random forest (RF) methods are used to classify the best split from the data and to identify the most important attributes to detect intrusion [25]. Intrusion DR (IDR) is considered as a fitness measure of the SSO-RF classifier, and the particle with highest IDR in the swarm is considered to be the best. At each iteration, attribute selection is performed by SSO, and within its loop, RF is used to classify the data. The SSO loop terminates when a stopping condition (the maximum number of iterations or 100% IDR) is reached. The evolutionary algorithm for SSO-RF is as follows [31]:

1. Let p be the population size, the maximum generation be MG, the maximum fitness function be MF, and the three predetermined constants be C_w, C_p, C_g.
2. Generate a random number $R = 0$ to 1 for d-dimension data.
3. Perform the comparison strategy where:
   If $(0 \le R < C_w)$, then $\{x_{nd} = x_{nd}\}$;
   Else if $(C_w \le R < C_p)$, then $\{x_{nd} = p_{nd}\}$;
   Else if $(C_p \le R < C_g)$, then $\{x_{nd} = g_{nd}\}$;
   Else if $(C_g \le R \le 1)$, then $\{x_{nd} = \text{new } (x_{nd})\}$;
4. RF $(n, d) = 1/\exp\ (x(n, d))$
   Update (x_{nd}).
5. Update pbest.
6. Update gbest.
7. Process will repeat until the stopping condition is satisfied or the maximum IDR is reached.

The experimental setup for intrusion detection is split into two steps. Initially, the SSO-RF algorithm is implemented to reduce the feature set for the IDS. Next, the result is validated using an RF classifier. The experiment is repeated ten times over both during the training and testing phases. The SSO-RF method is evaluated for 50 particles, and the iterations are repeated for 100 generations or until the maximum IDR is reached.

Table 2. Names of reduced attributes.

Proposed SSO-RF
1. Protocol type
2. Service name
3. Flag status
4. Size of source in bytes
5. Size of destination in bytes
6. Emergency flag
7. Connection status
8. Number of outbound commands
9. Accessed files count
10. Host login status
11. Guest login status

The proposed method filters the most suitable attributes based on the number of normal connections and reduces the dimensions of the live dataset. The eleven attributes (see Table 2) are further used in the fuzzy Wiener detection engine to analyze the intrusive behavior. The proposed method (SSO-RF) produces near optimal results compared to other intelligent swarm techniques [31].

2. Fuzzy Rule Generation Using Wiener Filter

In the proposed system, initially, the fuzzy rules are automatically generated using Lagrange interpolation with the successive approximation method. This is based on a sequence of rule approximations that converge on the solution and that are constructed recursively (iteratively); that is, each new rule is calculated on the basis of the preceding rule generation.

Let $f(x)$ be a continuous function in the interval $[a, b]$. To locate the position of the initial rule of the function $f(x) = 0$, we divide the interval $[a, b]$ into n subintervals as $a = x_0 < x_1 < x_2 < \cdots < x_n = b$, where $x_i = x_0 + ih, i = 0, 1, 2, \ldots, n, h = 0$. Let the rule generation equation be $x = f(x)$ for $x_1, \ldots, x_n, \ldots$, such that $x_1 = f(x_0)$, $x_2 = f(x_1)$, $x_3 = f(x_2)$ ... $x_n = f(x_{n-1})$. Iterate the same process until $(x_n - x_{n-1})$ is smaller than some specified tolerance (that is, the maximum IDR is reached). The result of the successive approximation is then processed using Lagrange interpolation to check whether the fuzzy rule belongs to one of seven linguistic values; attributes that fall within the seven linguistic values are further filtered using a Wiener filter to generate the data in terms of the normal or attack categories. The procedure for fuzzy rule generation using the Wiener filter is presented below.

A. Procedure for Fuzzy Rule Generation Using Wiener Filter to Detect Attacks

Step 1. An input dataset, D, is divided into five subsets as $D = \{D_i;$ where $i = 1 \le i \le 5\}$ based on a fuzzy predictor. The dataset is then sent as the input signal $x_i(n)$ to the Wiener filter.

Step 2. The Wiener filter is modeled as a linear combination of normal dataflow $s_i(n)$ and abnormal dataflow $\eta(n)$, where $x_i(n) = s_i(n) + \eta(n)$.

Step 3. The Wiener filter is used to remove abnormal data as the noise signal and extract the normal output data s_i.

Step 4. To achieve a balanced estimator, the fuzzy predictor uses seven linguistic variables (that is, VS, S, SM, M, LM, L, and VL) to generate rules to achieve better results.

Step 5. Based on the fuzzy rule, the Wiener filter is used as the error detector to classify the attack data in the network. The filter is broadly used to reduce the MMSE.

Step 6. The correlation matrix generated by the Wiener filter is used in the chi-squared test, which returns an output result of either 0 or 1.

Step 7. The result 0 indicates significant indifference (known as normal data), and the result 1 indicates significant difference, known as abnormal data.

Figure 3 shows the process flow for automatic fuzzy rule generation. The transaction values are used to identify security threats that have already been classified according to a given predefined intruder detection rule. A normal network transaction can easily pass through this process (the loop executed in our computer to check data); however, an ambiguous transaction will be suspended from subsequent phases. In the case of an ambiguous transaction, an alert event and inspection criteria are triggered. If there is a rule to identify the ambiguous transaction, then the numerical and logical values are adjusted and fine-tuned to identify any attack related to the ambiguous transaction. If there is no rule to identify the ambiguous transaction, then a new rule is generated based on logical and numerical values. If a new rule has to be generated, then merging this new rule with the existing reference database makes the existing numerical and logical values stored in our database prone to change following further similar transactions with similar input values; this is the essential aspect of the machine learning procedure that makes the proposed system impregnable to attacks. The reference database is frequently analyzed and updated to group the harmonic parameters (11 attributes) into a correlative group and the non-harmonic parameters into an auto-correlative group, in an accurate manner.

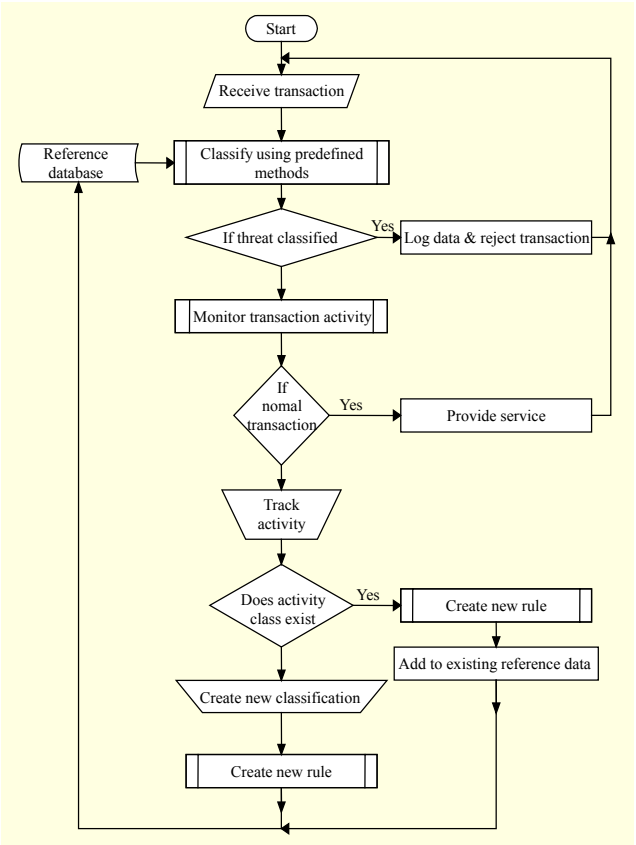

Fig. 3. Process flow of proposed work.

B. SSO Module

To improve the accuracy of the proposed method, SSO is used to evolve the optimal solution for the current population. The SSO algorithm repeatedly modifies the population based on the velocity of the particle moving from the *pbest* (personal) to the *gbest* (global) position.

In the proposed system, each individual particle is given as input to the MF of the fuzzy decision module. The new concept of the adaptive fuzzy granule fitness function is used for fitness generation to evaluate DR and FAR. The fitness queue is based on the insufficient similarity of the individual; if an individual is sufficiently similar to a known fuzzy granule, then that granule's fitness is used.

The fuzzy fitness can be calculated using

$$G = \{c_d, \sigma_d, L_d\},$$

$$\text{where } c_d \in R^m,\ \sigma_d \in R,\ L_d \in R,\ d = 1, 2, \ldots, l,$$

where initially $l = 0$, to make the fuzzy queue empty, C_d is an m-dimensional vector of centers, σ_d is the width of the MFs of the dth fuzzy granule, and L_d is the granule's life index. The granule G can be computed using

$$\overline{\mu}_{j,d} = \sum_{r=1}^{n} \frac{\mu_{d,m}(x_{j,m}^{i})}{n},$$

where $X_j^i = \{X_{j,1}^1, X_{j,2}^1, \ldots, X_{j,m}^1, \ldots, X_{j,n}^1\}$ is the ith individual in the jth particle. The fitness of X_j^i is computed

either by the exact fitness function or estimated fitness function by associating it to one of the granules in the queue with high similarity to X_j^i rather than a predefined threshold as follows:

$$f(X_j^i) = \frac{f(x_d) \text{ if } \max d \in \{1,2,\ldots,t\} \{\overline{\mu}_{j,d}\} > \theta_i}{f(X_j^i) \text{ computed by fitness function, otherwise}}. \quad (15)$$

Threshold θ_i increases as the best individual's fitness in generation i increases. Hence, as the particle matures, the fuzzy fitness queue reaches the highest fitness valuation.

V. Analysis of Experimental Results

The live dataset collected based on Table 1 is used as the test data to evaluate the classifier. The experimental analysis was performed using 30% of the training data, and the full dataset (see Table 1) was used for the testing phase to analyze the performance of the proposed system. The training dataset, which is based on fuzzy rule generation, trains the classifier, and the Wiener filter is used as an error detector; both the classifier and the Weiner filter are trained based on fuzzy rule generation. To make the proposed system more accurate, a fuzzy granule fitness function is used to generate various fitness functions based on similarity and an SSO optimizer is used to further optimize the ID. Table 3 below shows the various parameters, such as DR and FAR, the training time, and the testing time.

The soft computing method (fuzzy logic) within the IDS along with a Weiner filter helps the IDS to detect attacks more accurately. The proposed work is also compared with several machine learning techniques in Table 4, and it is found that the fuzzy rule generation with Wiener filter gives a strong performance in detecting various attacks, under the categories mentioned in this paper, in computer networks. Furthermore, this method is more flexible and convenient to this research work due to its automatic rule generation and adaptive fuzzy granule fitness queue. The DR and FAR are calculated based on the following:

$$\text{DR} = \frac{\text{Total number of correctly classified attacks}}{\text{Total number of instances}} \times 100, \quad (16)$$

$$\text{FAR} = \frac{\text{Total number of misclassified instances}}{\text{Total number of instances}} \times 100. \quad (17)$$

The current research work on intrusion detection gives the best DR only in relation to DOS and probe attacks rather than the other two; however, in real-time scenarios, it is U2R and R2L that play the greater role. This research work focuses not only on DOS and Probe attacks but also on U2R and R2L attacks in terms of both accuracy and FAR.

Table 3. Performance of proposed system.

Models	Class name	Accuracy (DR)	FAR	Train time (s) 30%	Test time (s) full dataset
Hybrid fuzzy Weiner method	Normal	88.72%	1.34%	71.2	203.0
	DOS	88.92%	1.34%	66.2	201.8
	Probe	87.92%	1.38%	71.2	203.0
	U2R	89.12%	1.32%	67.2	202.1
	R2L	89.14%	1.29%	67.5	202.2

Table 4. DR and FAR results for other machine learning techniques.

Techniques	DR	FAR
Fuzzy logic	74.5%	2.51%
SVM + fuzzy logic	80.7%	2.30%
Kalman filter + fuzzy logic	81.4%	1.97%
GA + fuzzy logic	83.5%	1.80%
Proposed fuzzy logic + Wiener filter	88.76%	1.34%

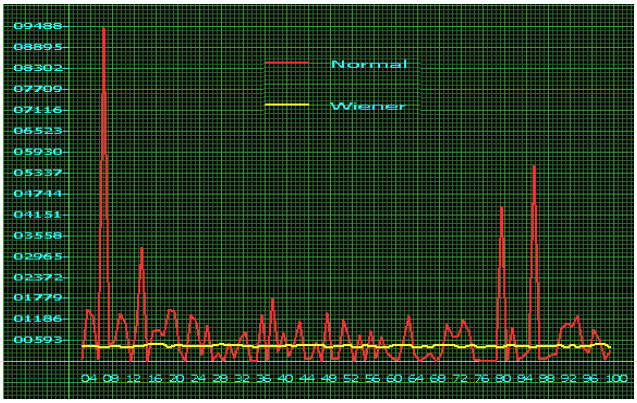

Fig. 4. Data sequence using normal and Wiener filter.

Figure 4 shows a data sequence using normal and Wiener filters, where the x-axis represents data sequence and the y-axis represents data size.

MSE values for individual attacks have also been experimented with in this paper, and Table 5 below shows individual MSE values both with and without the Wiener filter. From the above table, the MSE without the Wiener filter remains constant. While using the Wiener filter, MMSE values are reduced for all attacks, which increase the accuracy of the proposed system. Calculated MSE values for dataflow in our system are plotted as an input data sequence along the x-axis, and their data sizes are plotted along the y-axis, as shown in Fig. 5 (for individual attacks). Figure 5 indicates the dataflow values using the Wiener filter for a DOS attack.

http://dx.doi.org/10.4218/etrij.15.0114.0275

Table 5. MSE values for individual attacks.

Attacks	MSE value	
	Normal filter	Wiener filter
DOS	0.85	0.11
Probe	0.85	0.02
U2R	0.85	0.14
R2L	0.85	0.43

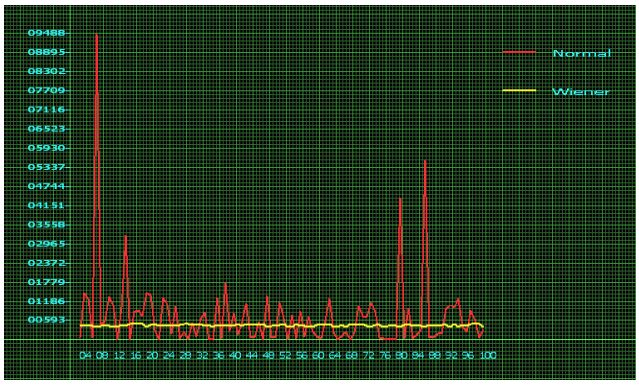

Fig. 5. MSE values for DOS attack.

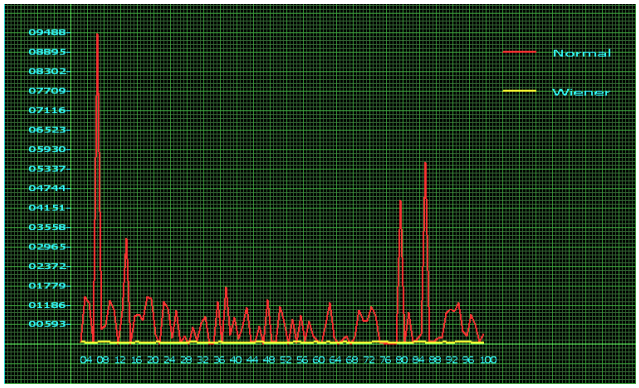

Fig. 6. MSE values for probe attack.

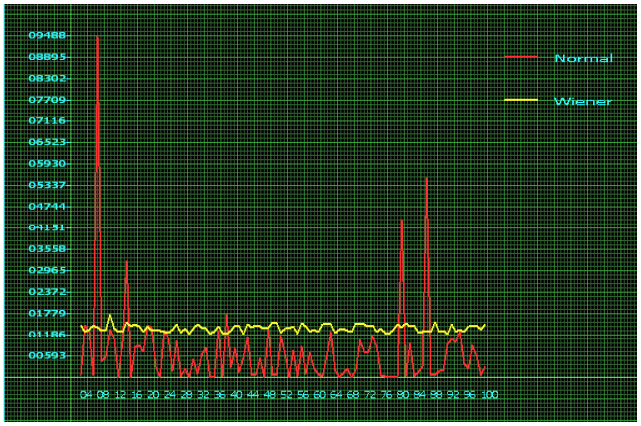

Fig. 7. MSE values for R2L attack.

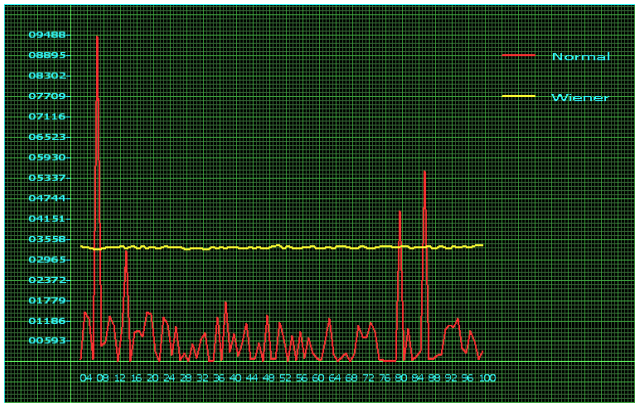

Fig. 8. MSE values for U2R attack.

The dataflow values reduce drastically for a probe attack (see Fig. 6). The proposed method works effectively and efficiently for U2R and R2L attacks. Figures 7 and 8 show the dataflow values for these attacks, respectively. From Figs. 7 and 8, it is clear from the yellow lines, which indicate the data flow from using a Wiener filter, that MSE values are reduced for all attacks.

VI. Conclusion

In this paper, automatic fuzzy rule generation combined with a Wiener filter is proposed for the live dataset and efficiently extracts numerous good rules for classification. In addition, simplified swarm optimization is used for optimization.

As an application of intrusion detection, an IDS was developed for real-time scenarios to detect attacks. The results obtained from the proposed method show that use of fuzzy rule generation combined with a Wiener filter in our IDS, and also in our database, works efficiently for detection both in simulation and in real time.

For intrusion detection, DR and FAR are considered as two important criteria for the developed IDS; as such, both of these are focused on in this research work along with training and testing times.

Initially, in the proposed system, the feature selection method, SSO-RF, reduces the number of attributes more effectively than the methods cited in [31], which makes the IDS more compact for detection. To assess the efficiency of the proposed system, we compared it against other machine learning techniques, and the results prove that the use of the Wiener filter in fuzzy rule generation increases the accuracy of intrusion detection.

In our proposed method, SSO is used to optimize the results obtained from the Weiner filter. The main advantage of SSO over other techniques is that there is no mutation or crossover required for processing individual populations, which may

reduce time-efficiency.

The important functions of the proposed system are that it efficiently extracts numerous fuzzy rules automatically and the Wiener filter uses statistical calculation based on a chi-squared test, both of which mean that the data is normal or abnormal.

The advantage of our automatic rule generation method is that it generates numerous rules based on real-time attacks; moreover, it automatically calculates deviations for normal connections. Our future work might focus on using various neural network techniques combined with fuzzy logic for detecting normal and intrusive data in a network.

References

[1] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Department of Computer Engineering, Chalmers University, 2000, Report no. 99-15.

[2] A. Lazarevic et al., "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," *Proc. SIAM Conf. Data Mining*, University of Minnesota, Minneapolis, MN, USA, 2003.

[3] S.-J. Han and S.-B. Cho, "Evolutionary Neural Networks for Anomaly Detection Based on the Behavior of a Program," *IEEE Trans. Syst., Man, Cybern. Part B*, vol. 36, no. 3, June 2005, pp. 559–570.

[4] W. Lu and I. Traore, "Detecting New Forms of Network Intrusion Using Genetic Programming," *Comput. Intell.*, vol. 20, no. 3, Aug. 2004, pp. 475–494.

[5] L.A. Zadeh, "Role of Soft Computing and Fuzzy Logic in the Conception, Design and Development of Information/Intelligent Systems," in *Comput. Intell.: Soft Comput. Fuzzy-Neuro Integr. Appl.*, Berlin, Germany: Springer Berlin, Heidelberg, 1998, pp. 1–9.

[6] A. Abraham and R. Jain, "Soft Computing Models for Network Intrusion Detection Systems," *Classification Clustering Konowl. Discovery*, Berlin, Germany: Springer Berlin Heidelberg, vol. 16, 2005, pp. 191–207.

[7] J. Gomez and D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection," *Proc. IEEE Workshop Inf. Assurance*, United States Military Academy, West Point, NY, USA, 2001, pp. 68–75.

[8] Y. Li and L. Guo, "An Active Learning Based TCM-KNN Algorithm for Supervised Network Intrusion Detection," *Comput. Security*, vol. 26, no. 7, Dec. 2007, pp. 459–467.

[9] H.A. Nguyen and D. Choi, "Application of Data Mining to Network Intrusion Detection: Classifier Selection Model," *Proc. Asia-Pacific Symp. Netw. Operation Manag.*, vol. 5297, 2008, pp. 399–408.

[10] Z. Zhang et al., "HIDE: A Hierarchical Network IDS Using Statistical Preprocessing and Neural Network Classification," *Proc. IEEE Workshop Int. Assurance Security*, West Point, NY, USA, 2001, pp. 85–90.

[11] S.T. Powers and J. He, "A Hybrid Artificial Immune System and Self Organizing Map for Network Intrusion Detection," *Inf. Sci.*, vol. 178, no. 15, Aug. 15, 2008, pp. 3024–3042.

[12] J.R. Koza, *Genetic Programming: On the Programming of Computing by Means of Natural Selection*, Cambridge, MA, USA: MIT Press, 1992.

[13] K. Shafi and H.A. Abbass, "Biologically-Inspired Complex Adaptive Systems Approaches to Network Intrusion Detection," *Inf. Security, Techn. Report*, vol. 12, no. 4, 2007, pp. 209–217.

[14] T. Sousa, A. Silva, and A. Neves, "Particle Swarm Based Data Mining Algorithms for Classification Tasks," *Parallel Comput.*, vol. 30, no. 5–6, May–June 2004, pp. 767–783.

[15] S. Revathi, *Linkware Technologies Private Limited*, Network Simulator Capture (NSC) Dataset. Accessed Dec. 18, 2013. https://www.linkware.in/

[16] Lincoln, *Darpa Intrusion Detection Dataset*. Accessed Jan. 11, 2011. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[17] UCI KDD Archive, *KDDCup99 Data*, University of California. Accessed Oct. 28, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[18] M. Tavallaee et al., "A Detailed Analysis of the KDDCup99 Data Set," *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, July 8–10, 2009, pp. 1–6.

[19] NSL-KDD Data Set, *Network-Based Idss*. Accessed Mar. 2009. http://nsl.cs.unb.ca/KDD/NSLKDD.html

[20] A.N. Toosi and M. Kahani, "A New Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers," *Comput. Commun.*, vol. 30, no. 10, July 31, 2007, pp. 2201–2212.

[21] Y.Y. Chung and N. Wahid, "A Hybrid Network IDS Using Simplified Swarm Optimization (SSO)," *Appl. Soft Comput.*, vol. 12, no. 9, Sept. 2012, pp. 3014–3022.

[22] S. Mabu et al., "An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining Using Genetic Network Programming," *IEEE Trans. Syst., Man., Cybern., — Part C: Appl. Rev.*, vol. 41, no. 1, Jan. 2011, pp. 130–139.

[23] S. Zaman, M. El-Abed, and F. Karray, "Features Selection Approaches for IDSs Based on Evolution Algorithms," *Int. Conf. Ubiquitous Inf. Manag. Commun.*, Kota Kinabalu, Malaysia, Jan. 17–19, 2013.

[24] M. Davarynejad, T. Akbarzadeh, and N. Pariz, "A Novel General Framework for Evolutionary Optimization: Adaptive Fuzzy Fitness Granulation," *Proc. IEEE Int. Conf. Evolutionary Comput.*, Singapore, Sept. 25–28, 2007, pp. 951–956.

[25] A.J. Malik, W. Shahzad, and F.A. Khan, "Network Intrusion Detection Using Hybrid Binary PSO and Random Forests Algorithm," *IEEE Congress Evolutionary Comput.*, New Orleans,

LA, USA, June 2011, pp. 662–668.
[26] M. Al-Kasassbeh, "Network Intrusion Detection with Wiener Filter-Based Agent," *World Appl. Sci. J.*, vol. 13, no. 11, 2011, pp. 2372–2384.
[27] L.A. Zadeh, "Fuzzy Sets," *Inf. Contr.*, vol. 8, no. 3, June 1965, pp. 338–353.
[28] B. Mulgrew, P. Grant, and J. Thompson, *Digital Signal Process. - Concepts and Applications*, Houndmills and London: Macmillan Press Ltd., 1999, pp. 1–408.
[29] S. Haykin, *Adaptive Filter Theory*, Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996, pp. 1–989.
[30] M.H. Hayes, *Statistical Digital Signal Process and Modeling*, New York, NY, USA: John Wiley & Sons, Inc., 1996, pp. 1–624.
[31] S. Revathi and A. Malathi, "Feature Extraction Using the Sim-Swadorest Optimization Algorithm for Intrusion Detection," *Int. Conf. Recent Innovations Comput. Sci. Inf. Technol.*, Singapore, Nov. 8, 2014, pp. 75–79.

Revathi Sujendran received her MSc degree in computer science from St. Joseph College of Arts and Science, Tamilnadu, India, in 2008 and her MPhil degree in computer science from Bharathidasan University, Tamilnadu, India, in 2009. She is now currently pursuing her PhD degree at PG and Research, Department of Computer Science, Government Arts College, affiliated to Bharathiar University, Tamilnadu, India. Her current research interests include network security, data mining, and computational intelligence.

Malathi Arunachalam received her MSc degree in computer science from Bharathidasan University, Tamilnadu, India, in 1991 and her MPhil degree in computer science from Bharathiar University, Tamilnadu, India, in 2002. She received her PhD in computer science from Bharathiar University, Tamilnadu, India, in 2012. She is currently serving as an assistant professor in PG and Research, Department of Computer Science, Government Arts College (Autonomous), Coimbatore, India. She has successfully completed one funded project sponsored by UGC. She has designed and developed many computer awareness programs for government school students. Her research interests include data mining, networking, and object-oriented programming.