Flow-Based Anomaly Detection Using Neural Network Optimized with GSA Algorithm

Zahra Jadidi, Vallipuram Muthukumarasamy, Elankayer Sithirasenan
School of Information and Communication Technology
Griffith University
Gold Coast, Australia
Email: zahra.jadidi@griffithuni.edu.au,
{v.muthu, e.sithirasenan}@griffith.edu.au

Mansour Sheikhan
Electrical Engineering Department, Engineering Faculty
Islamic Azad University, South Tehran Branch
Tehran, Iran
Email: msheikhn@azad.ac.ir

Abstract—Reliable high-speed networks are essential to provide quality services to ever growing Internet applications. A Network Intrusion Detection System (NIDS) is an important tool to protect computer networks from attacks. Traditional packet-based NIDSs are time-intensive as they analyze all network packets. A state-of-the-art NIDS should be able to handle a high volume of traffic in real time. Flow-based intrusion detection is an effective method for high speed networks since it inspects only packet headers. The existence of new attacks in the future is another challenge for intrusion detection. Anomaly-based intrusion detection is a well-known method capable of detecting unknown attacks. In this paper, we propose a flow-based anomaly detection system. Artificial Neural Network (ANN) is an important approach for anomaly detection. We used a Multi-Layer Perceptron (MLP) neural network with one hidden layer. We investigate the use of a Gravitational Search Algorithm (GSA) in optimizing interconnection weights of a MLP network. Our proposed GSA-based flow anomaly detection system (GFADS) is trained with a flow-based data set. The trained system can classify benign and malicious flows with 99.43% accuracy. We compare the performance of GSA with traditional gradient descent training algorithms and a particle swarm optimization (PSO) algorithm. The results show that GFADS is effective in flow-based anomaly detection. Finally, we propose a four-feature subset as the optimal set of features.

Keywords—Flow-based anomaly detection, Multi-layer Perceptron, Gravitational Search Algorithm.

I. INTRODUCTION

While our dependence on the Internet is increasing, a wide range of vulnerabilities has caused growing interest in intrusion detection system (IDS). There are two approaches in IDS: anomaly-based and misuse-based methods. The anomaly-based category detects activities deviating from established patterns for network traffic. This method is able to detect unknown attacks but it cannot distinguish between different types of attacks. On the other hand, misuse-based IDS compares users activities with pre-defined signatures [1]. This IDS is very accurate in detecting known attacks but it cannot detect unknown attacks. Network-based IDS (NIDS) monitors data exchanged between computers over a network [1]. In this study, we propose an anomaly-based NIDS. Traditional NIDS were packet-based and inspected every packet. Recent high speed networks put a heavy computational load on traditional NIDS. Examining all packets captured in the network is very time-consuming [2]. Nowadays, researchers address the problem of anomaly detection in high speed network by flow-based traffic analysis. A flow is a group of IP packets with some common properties passing a monitoring point in a specified time interval [3]. Flow-based monitoring is based on information in packet headers. NetFlow is a Cisco proprietary protocol. It is enabled on router devices to provide network flows. Flow-based NIDS processes lower traffic in comparison with packet-based NIDS. For instance, at the University of Twente network, the ratio between packets exported by NetFlow and packets on the network was 0.1 [4]. This confirms that flow-based NIDS is scalable, meaning that it can be used in high-speed networks. Flow-based data also decreases privacy concerns in comparison with a packet-based data set due to the absence of payload. A comprehensive survey is provided in [5] about flow-based intrusion detection. Different anomaly detection methods have been applied in traditional intrusion detection, for example, statistical methods, artificial neural networks (ANN), data-mining methods and support vector machines (SVM) [6]. A feed-forward neural network (FFNN) is a well-known method in anomaly detection. MLP is a FFNN. A MLP with two layers is suitable for classification of input patterns. Training is very important in ANN since it is responsible for adjusting weights and biases to minimize error. The back-propagation (BP) is a well-known method to train MLPs. BP is a gradient-descent technique. The main drawback of this method is that it tends to converge into local minima. In recent years, many heuristic algorithms have been successfully applied to solve the problem of gradient-descent techniques, for example, Genetic Algorithm, Simulated Annealing, Ant Colony Search Algorithm and Particle Swarm Optimization (PSO) [7]. Gravitational Search Algorithm (GSA) is one of the newest swarm based heuristic algorithms proposed in [7]. This algorithm is based on the Newtonian gravity. GSA is proposed to overcome the slow convergence and local minima problem in BP (gradient descent technique) [8]. In addition, GSA has merits compared to some well-known algorithms like PSO. PSO uses memory to update the velocity while GSA is memory-less and its updating procedure is based on the current position of the agents. GSA has a faster convergence in comparison with PSO. Also, GSA has an adaptive learning rate [7]. An algorithm based on GSA and a heuristic search algorithm (GSA-HS) is proposed to improve clustering of the tested data set in [9]. A prototype classifier based on GSA is used in [10] to classify instances in multi-class data sets and paper compares the result with other algorithms like PSO and Artificial Bee Colony (ABC). The results show the
effectiveness of a GSA based classifier in resolving classification problems. Different flow-based NIDS have been proposed. An online DoS resilient flow-level intrusion detection system for high-speed network called HiFIND is proposed in [11]. The paper uses sketches to detect anomalies. A sketch is a one-dimensional hash table that is good for storing information. HiFIND has high accuracy and low memory usage. The combination of data mining and visualization is used in [12] to provide a flow-based Botnet detection system with a good performance. Statistical techniques are used for anomaly detection in network flows in a controlled environment [13]. This environment represents consistent patterns of network activities. This can be used for the prediction of future activities. Any deviation from the forecast shows anomaly. The results show that the proposed method can be used in real-time anomaly detection. Alshammari and Zincir-Heywood [14] use five learning algorithms like SVMs and C4.5 to classify encrypted traffic using flow-based features. SSH and Skype are chosen as encrypted traffic. The results show that C4.5 has the best performance. While bench mark data sets are essential, studies suffer from lack of public data sets for evaluating their proposed flow-based NIDS. The first labeled data set including network flows is provided in [15]. Winter uses the modified version of this data set to train one-class SVM (OC-SVM) to classify the incoming traffic in two groups: In-class (malicious) or outlier (not malicious) [2]. However, attacks related to packet payload cannot be detected with flow-based NIDS. Some attacks such as DoS attacks, Worms, Scans and Botnets can be easily detected by flow-based NIDS [5]. DoS attacks, which overload the network is easily detectable by this method. On the other hand, semantic DoS attack that is related to payload contents are impossible to be detected by flow-based method. An overview of flow-based and packet-based intrusion detection performance in high speed networks is provided in [16]. This study introduces flow-based NIDS as a logical choice for high-speed networks. Nevertheless, a flow-based NIDS is not a replacement for a packet-based NIDS. In an ideal world, the attacks that can be detected by a flow-based NIDS should be also detectable by a packet-based NIDS. However, in the real world this is not correct for high speed networks mainly because of limitations like the high resource requirement of packet-based NIDS. These different NIDSs can cover different categories of attacks in the real word [17]. Flow-based anomaly detection is considered in our study in which we use the GSA algorithm to optimize the interconnection weights of the two layers MLP. This optimized MLP is trained with a flow-based data set to distinguish between benign and malicious traffic. In our study, we use the modified version of the first labeled flow-based data set for evaluation of our proposed GSA-based flow anomaly detection system (GFADS). This modified data set, proposed by Winter [2], includes flows defined according to NetFlow version 5. In this NetFlow, a flow has the following features [15]: Source/Destination IP address, Source/Destination ports, the level 3 protocol type, total number of packets transferred in a flow, the amount of transferred octets in a flow, the TCP header flags and the start and end time of the flow. The remainder of this paper is organized as follows. Section II explains the data set used. Section III provides a brief description of algorithms and our proposed GSA-based flow anomaly detection system. Section IV provides the experimental results of GFADS and discusses the proposed solutions. Section V concludes the paper.

II. DATA SET

Sperotto, Sadre, Vliet and Pras [15] provide the first public labeled flow-based data set. The data set was captured in the university of Twente network by monitoring a honeypot. A honeypot is a vulnerable environment used to observe attacks. Three services were installed on this host: SSH (OpenSSH) service, FTP and Apache web server. The honeypot connected to Internet for a period of 6 days. Sperotto’s data set [15] is divided into three categories: Malicious traffic, Side-effect traffic (this part is not by itself malicious, for example, ICMP, Auth/Ident and IRC traffic), Unknown traffic and uncorrelated alerts (in this part, the malicious or benign nature of the traffic cannot be identified). The large number of flows in this data set makes the training phase time-consuming. To address this problem, Winter [2] provides the modified version of data set. For this modification, all side-effect flows, unlabeled flows, duplicate flows and all protocols other than SSH and HTTP are deleted. Afterwards, around 20,000 samples are randomly chosen among the remaining data set then succeed flows are added to them. The new modified data set contains 22,924 malicious flows. In terms of number of features, there is a change in the modified version. Each flow contains 10 features in the Sperotto’s data set while this is decreased to seven features in Winter’s data set [2], shown in Table 1. IP addresses are deleted because they have been anonymized. Furthermore, features related to time are also eliminated and a feature called duration is added. All collected flows in Sperotto’s data set are related to malicious activities while the benign flows are also needed for measuring the performance [2]. To generate the required benign traffic in Winter’s data set, a tcpdump is used to capture network traffic then a host extracts flows. Each benign flow has seven features. The benign data set contains HTTP, SSH, DNS, ICMP and FTP. In our study, Winter’s data sets are used for training and testing of GFADS. Table 2 presents the detail of training and testing data sets.

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets/flow (P)</td>
<td>Transferred packets in the flow.</td>
</tr>
<tr>
<td>Octets/flow (O)</td>
<td>Transferred octets in the flow.</td>
</tr>
<tr>
<td>Duration (D)</td>
<td>Duration of the flow (in seconds).</td>
</tr>
<tr>
<td>Source port (SP)</td>
<td>Source port of the flow.</td>
</tr>
<tr>
<td>Destination port (DP)</td>
<td>Destination port of the flow.</td>
</tr>
<tr>
<td>TCP flags (TF)</td>
<td>TCP flags in the flow.</td>
</tr>
<tr>
<td>IP protocol (PR)</td>
<td>IP protocol number of the flow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data set</td>
<td>962</td>
</tr>
<tr>
<td>Testing data set</td>
<td>942</td>
</tr>
<tr>
<td>Entire traffic</td>
<td>1904</td>
</tr>
</tbody>
</table>

Table 1. Winter’s data set features

Table 2. Construction of training and testing data set
III. ALGORITHMS AND PROPOSED SYSTEM

Two optimization algorithms used in this study and the proposed system are described in this section.

A. Gravitational Search Algorithm

According to Newton law, each particle in the universe attracts other particles. This is the basis of the GSA algorithm [7]. In GSA, agents are considered as objects and their masses are used for measuring their performance. All objects move towards the objects with heavier masses. Heavy masses move more slowly and are known as good solutions. There are three kinds of masses [7]. \( M_p \) is active gravitational mass which shows the strength of the gravitational field because of a particular object. \( M_t \) is passive gravitational mass which is related to the strength of an object’s interaction with the gravitational field. Inertial mass or \( M_i \) is related to the resistance of an object to changing its state of motion when a force is applied. The agents with bigger inertia mass have slower motion in search space and a more accurate search while the bigger gravitational mass has faster convergence due to having a higher attraction. Each mass (agent) in GSA also has position corresponding to the solution of a problem. Also, a fitness function is used to determine the gravitational and inertial masses [7]. In (1), the position of \( i \)-th agent in the \( d \)-th dimension is \( x_{id}^i \).

\[
X_i = (x_{i1}^1, x_{i2}^1, \ldots, x_{id}^i, \ldots, x_{iN}^i) \quad \text{for} \quad i = 1, 2, \ldots, N \tag{1}
\]

In (2), \( F_{ij}^d(t) \) is the force on mass 'i' from mass 'j' at time 't'. \( M_{ij} \) is the active gravitational mass of agent j. The passive gravitational mass of agent i is \( M_{pi} \). Gravitational constant at time t is \( G(t) \). \( \varepsilon \) is a small constant and \( R_{ij}(t) \) shows the Euclidian distance between agents i and j. It is defined as in (3).

\[
F_{ij}^d(t) = G(t) \frac{M_{ij}(t) \times M_{pi}(t)}{R_{ij}(t) + \varepsilon} (x_{jd}^j(t) - x_{id}^i(t)) \tag{2}
\]

\[
R_{ij}(t) = \|X_i(t) - X_j(t)\|_2 \tag{3}
\]

\( rand_i \) shows a random number in the interval [0, 1]. The total force on agent i is as is in (4). The acceleration of the agent i in direction \( d \) is defined as in (5). The next velocity and position of an agent are defined as in (6) and (7).

\[
F_i^d(t) = \sum_{j=1}^{N} M_{pi} rand_j F_{ij}^d(t) \tag{4}
\]

\[
a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{5}
\]

\[
v_i^d(t + 1) = v_i^d(t) + a_i^d(t) \tag{6}
\]

\[
x_i^d(t + 1) = x_i^d(t) + v_i^d(t + 1) \tag{7}
\]

G, the gravitational constant, is \( G_0 \) at the beginning and will be decreased with the time. In GSA, G is defined by (8), where \( \alpha \) is a constant. T is the total number of iterations and t is the current iteration.

\[
G(t) = G_0 e^{-t/T} \tag{8}
\]

The inertial mass is defined as in (9) while \( fit_i(t) \) is the fitness value of agent i. The best(t) and worst(t) for minimization problems are defined by (10) and (11) and for maximization they are defined by (12) and (13):

\[
m_i(t) = \frac{fit_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \tag{9}
\]

\[
\text{best}(t) = \min_{j \in \{1, \ldots, N\}} fit_j(t) \tag{10}
\]

\[
\text{worst}(t) = \max_{j \in \{1, \ldots, N\}} fit_j(t) \tag{11}
\]

\[
\text{best}(t) = \max_{j \in \{1, \ldots, N\}} fit_j(t) \tag{12}
\]

\[
\text{worst}(t) = \min_{j \in \{1, \ldots, N\}} fit_j(t) \tag{13}
\]

In GSA, only Kbest agents attract other agents. Kbest has the initial value of \( K_b \) but it is reduced with time. At first, all agents apply the force and finally only one agent remains applying force to the others so we will have (14) instead of (4).

\[
F_i^d(t) = \sum_{j \in K\text{best}(t)} \text{rand}_j F_{ij}^d(t). \tag{14}
\]

We can summarize the function of GSA as follows [7]:
1) Initialze population. 2) Fitness evaluation for each agent. 3) Update \( G(t) \), best(t) and worst(t). 4) Calculate \( M \) and acceleration for agents. 5) Calculate the new velocity and position. 6) If meet end of criterion, it is the best solution else go to step 2. The comparison between GSA and PSO shows that GSA has merit in the field of optimization [7].

B. Particle Swarm Optimization algorithm

The PSO algorithm, proposed by Kennedy and Eberhart [18], has a population based search procedure in which particles (individuals) change their position (state) with time. Particles of PSO algorithm fly around in a multidimensional search space. Each particle changes its position during flight. These changes are according to its own experience and neighbouring particle to use the best position encountered by itself and its neighbour. The position of each particle is as in (15) [19]:

\[
x_i(k + 1) = x_i(k) + v_i(k + 1) \tag{15}
\]

\( k \) is the discrete time index. \( v_i(t) \) is the velocity and \( x_i \) is the position of \( i \)-th particle. In our simulations, (16) is used for velocity [19]:

\[
v_i(k + 1) = \varphi(k) v_i(k) + \alpha_1 [y_{1i}(P_i - x_i(k))] + \alpha_2 [y_{2i}(G - x_i(k))] \tag{16}
\]

\( P_i \) is personal best which is the best position found by \( i \)-th particle and \( G \) is global best which is the best position found by swarm. \( y_{1i} \) and \( y_{1i} \) are two random numbers applied to \( i \)-th particle. These numbers are in the interval [0,1]. \( \alpha_1 \) and \( \alpha_2 \) are the acceleration constants. \( \varphi(k) \) is the inertia function. Linear decreasing strategy is used in this paper so an initially large inertia weight is linearly decreased to a small value as in (17):

\[
\varphi(k) = [\varphi(0) - \varphi(N_p)] \frac{(N_R - k)}{N_R} + \varphi(N_R) \tag{17}
\]

\( N_R \) shows the maximum number of time steps. The initial inertia weight is \( \varphi(0) \) and the final inertia weight is \( \varphi(N_R) \). The steps of PSO algorithm are summarized as follows: Step 1: Generate initial swarm and random position and velocity of
each particle \((x_i, v_i; i = 1, ..., M)\). Step 2: Calculate the fitness function for each particle \((f(y_i) = fitness(x_i))\). Step 3: Generate initial value for each \(P_i\) and \(G\) as \(P_i = y_i\) and \(G = min\{P_i\}; i = 1, ..., M\). Step 4: Calculate the new velocity of each particle by dynamic inertia weight ((16) and (17)) and use velocity clamping as in (18) to control it. Update the position (16). Step 5: Update \(P_i\) and \(G\). \(y_i, new = fitness(x_i, new)\) \(P_i = y_i, new\) and \(G = min\{P_i\}\). Step 6: If it meets the stop conditions, stop and return \(G\) as the best solution else go to Step 4.

\[
v_{i}(k + 1) = \begin{cases} v_{i}(k + 1) & \text{if } v_{i}(k + 1) < V_{max} \\ V_{max} & \text{if } v_{i}(k + 1) \geq V_{max} \end{cases} \tag{18}
\]

C. Proposed GSA-based flow anomaly detection system

The classification of nonlinearly separable patterns can be performed using MLP with two layers. MLP is commonly trained with BP algorithms. In BP algorithms, sometimes the convergence is to the points that are the best solutions locally (local minima) not globally (global minimum). To avoid local minima, heuristic optimization algorithms have been proposed. GSA is a novel heuristic optimization algorithm. It normally works well in searching for the global minimum. In our proposed GFADS, we first employ a GSA algorithm to optimize the interconnection weights of a two-layer MLP and then the optimized MLP is deployed to detect anomalies in a flow-based traffic. The two-layer MLP has one hidden layer and an output layer. Fig. 1 shows the structure of GFADS. The MLP has three nodes in the hidden layer and two nodes in the output layer. Two output nodes perform the classification of the flow-based traffic into malicious and benign subsets. The parameters of the GSA algorithm are shown in Table 3. Three nodes in the hidden layer and the selected GSA parameters gave the best performance. We apply the Winter’s data set to train our GFADS. Each flow in the training dataset has seven features. We implemented our system in MATLAB version R2012a (7.14.0.739). After training, the optimized MLP is able to detect new and unknown attacks. However, it cannot be trained with the data set in its original form therefore pre-processing is required. The data set should be scaled to the range [-1; +1] to achieve optimal classification results. We use the Min-Max normalization method performing as given in (19) [20]:

\[
x’ = (t_{max} - t_{min}) \times \frac{(x - x_{min})}{(x_{max} - x_{min})} + t_{min} \tag{19}
\]

\(x_{max}\) and \(x_{min}\) are the maximum and minimum values of each feature. The data is rescaled to the range of values \((t_{max}, t_{min})\).

IV. EXPERIMENTAL RESULTS AND DISCUSSION

For measuring the performance, four metrics are used [2]. Accuracy, Error Rate (ER), Miss Rate (MR) and False Alarm Rate (FAR) are defined as Table 4. True Positive \(t_p\) and True Negative \(t_n\) show correct detection of malicious and benign traffic respectively. False Positive \(t_f\) corresponds to wrong detection of benign traffic and False Negative \(t_{fa}\) is the error in the detection of malicious traffic. Limiting FAR in anomaly detection is a priority [2]. The impact of the number of masses and hidden layer nodes on FAR are investigated in our study. Five masses, three nodes in hidden layer and other parameters in Table 3 give the lowest FAR. GFADS is trained with pre-processed training data set and is evaluated in classifying the testing flows into malicious or benign subsets. Data sets contain flows with seven features. To compare GSA with some other training algorithms in optimization of MLP weights, five gradient descent algorithms are evaluated. Updating weight and bias values are carried out according to gradient descent momentum and an adaptive learning rate in traingdx. Gradient descent backpropagation (traingd), Gradient descent with adaptive learning rate backpropagation (traingda), Gradient descent with momentum backpropagation (traingdm) and sequential order incremental training with learning function (trains) are used to update weights. PSO algorithm is also deployed to train MLP. The parameters of PSO are shown in Table 3. The performance of GSA and these training algorithms in tuning MLP weights are investigated and the experiments are repeated for 10 times. Table 5 shows the averaged results. Iteration number is 600 for the whole networks. According to Table 5, GSA creates the highest accuracy which is comparable with Traingdx and PSO. Additionally, GSA has lowest FAR.

![Image](image-url)
The maximum and minimum values of accuracy of these algorithms over 10 experiments are reported in Table 6. For cross validation, we divide our testing data set to two separate data sets (called data set 1 and data set 2) each of which has equal benign and malicious flows. Table 7 compares the performance of GFADS when it is tested by different test data sets. The results are the averaged results of 10 experiments. According to this table, GFADS has high accuracy in all data sets. To investigate the performance of GFADS, we repeat our examinations with small numbers of iteration. Fig 2 represents the results. GSA has better accuracy in low iterations compared to the gradient descent based algorithms. According to these results, GSA achieves to the desired performance earlier than the others and it converges faster. Having a limited FAR is very important in anomaly detection. This is a priority in [2] and feature selection is done based on that. Flow-based intrusion detection by using One-Class SVM is implemented in [2]. Different parameters of SVM and different feature subsets are examined and a subset containing the following four features is proposed in [2]: source port (SP), destination port (DP), TCP flags (TF) and IP protocol (PR). In our study, we also test the effect of different subsets of seven features on the error rates of GFADS. The accuracy of GSA is approximately stable after 100 iterations. In this step, 150 is chosen for iteration number to find the best subset. Table 8 represents the subsets generating low FAR. According to this table, GFADS highly depends on SP and DP. The set including Octets/flow (O), SP, DP and TF gives the lowest FAR which is close to the FAR of the subset proposed in [2].

We selected this new subset as the best features for our GFADS. The results are approximately equal to seven-feature based results. Table 9 compares the performance of GFADS and OC-SVM based NIDS trained with SP, DP, TF and PR features. OC-SVM [2] trained with only malicious flows. This affects the FAR. The parameters of OC-SVM are adjusted to achieve zero FAR [2] but it causes lower accuracy in comparison with GFADS. The training set in our study have both malicious and benign traffic. The results show that OC-SVM gives lower FAR and higher MR than GFADS. High MR shows a lot of intrusions go undetected. FAR leads NIDS to generate false alarms. Therefore, both of them should be low.

![Figure 2. Comparison of training algorithms with different Iteration numbers](image)

Table 5. Performance metrics of GFADS and other algorithms

<table>
<thead>
<tr>
<th>Features</th>
<th>Train Gdx</th>
<th>Train gd</th>
<th>Train gdm</th>
<th>Trains</th>
<th>GSA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>tn</td>
<td>933</td>
<td>233</td>
<td>429</td>
<td>873</td>
<td>936</td>
<td>915</td>
</tr>
<tr>
<td>tp</td>
<td>7640</td>
<td>7678</td>
<td>7560</td>
<td>7666</td>
<td>7650</td>
<td>7645</td>
</tr>
<tr>
<td>fp</td>
<td>9</td>
<td>709</td>
<td>9</td>
<td>513</td>
<td>69</td>
<td>6</td>
</tr>
<tr>
<td>fn</td>
<td>48</td>
<td>10</td>
<td>128</td>
<td>22</td>
<td>38</td>
<td>43</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>99.33</td>
<td>91.67</td>
<td>98.41</td>
<td>93.80</td>
<td>98.76</td>
<td>99.43</td>
</tr>
<tr>
<td>ER (%)</td>
<td>0.66</td>
<td>8.3</td>
<td>1.59</td>
<td>6.20</td>
<td>1.24</td>
<td>0.56</td>
</tr>
<tr>
<td>MR (%)</td>
<td>0.62</td>
<td>0.13</td>
<td>1.67</td>
<td>0.29</td>
<td>0.49</td>
<td>0.56</td>
</tr>
<tr>
<td>FAR (%)</td>
<td>0.96</td>
<td>75.27</td>
<td>0.96</td>
<td>54.46</td>
<td>7.33</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table 6. Accuracy of GFADS and other algorithms over 10 experiments

<table>
<thead>
<tr>
<th>Features</th>
<th>Minimum Accuracy</th>
<th>Maximum Accuracy</th>
<th>Average of accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traingdx</td>
<td>98.52</td>
<td>99.49</td>
<td>99.33</td>
</tr>
<tr>
<td>Traingd</td>
<td>90.09</td>
<td>93.24</td>
<td>91.67</td>
</tr>
<tr>
<td>Traingda</td>
<td>96.42</td>
<td>99.47</td>
<td>98.41</td>
</tr>
<tr>
<td>Traingdm</td>
<td>91.12</td>
<td>94.09</td>
<td>93.80</td>
</tr>
<tr>
<td>Trains</td>
<td>96.32</td>
<td>98.95</td>
<td>98.76</td>
</tr>
<tr>
<td>GSA</td>
<td>98.90</td>
<td>99.63</td>
<td>99.43</td>
</tr>
<tr>
<td>PSO</td>
<td>97.69</td>
<td>99.52</td>
<td>99.38</td>
</tr>
</tbody>
</table>

Table 7. Performance of GFADS with different testing sets

<table>
<thead>
<tr>
<th>Testing data set</th>
<th>Accuracy (%)</th>
<th>Correct</th>
<th>Incorrect</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole test data set</td>
<td>99.43</td>
<td>8581</td>
<td>49</td>
<td>8630</td>
</tr>
<tr>
<td>Data set 1</td>
<td>99.47</td>
<td>4292</td>
<td>23</td>
<td>4315</td>
</tr>
<tr>
<td>Data set 2</td>
<td>99.30</td>
<td>4285</td>
<td>30</td>
<td>4315</td>
</tr>
<tr>
<td>Training dataset</td>
<td>99.64</td>
<td>16140</td>
<td>58</td>
<td>16198</td>
</tr>
</tbody>
</table>

Table 8. GFADS trained with different features subsets

<table>
<thead>
<tr>
<th>Features</th>
<th>tn</th>
<th>tp</th>
<th>fp</th>
<th>fn</th>
<th>MR (%)</th>
<th>FAR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP, DP, TF</td>
<td>930</td>
<td>7642</td>
<td>12</td>
<td>46</td>
<td>0.60</td>
<td>1.27</td>
</tr>
<tr>
<td>O, SP, DP</td>
<td>933</td>
<td>7642</td>
<td>9</td>
<td>46</td>
<td>0.60</td>
<td>0.96</td>
</tr>
<tr>
<td>D, SP, DP</td>
<td>921</td>
<td>7652</td>
<td>21</td>
<td>36</td>
<td>0.47</td>
<td>2.23</td>
</tr>
<tr>
<td>SP, DP, TF, PR</td>
<td>934</td>
<td>7642</td>
<td>8</td>
<td>46</td>
<td>0.60</td>
<td>0.85</td>
</tr>
<tr>
<td>P, SP, DP, PR</td>
<td>933</td>
<td>7642</td>
<td>9</td>
<td>46</td>
<td>0.60</td>
<td>0.96</td>
</tr>
<tr>
<td>O, SP, DP, TF</td>
<td>935</td>
<td>7643</td>
<td>7</td>
<td>45</td>
<td>0.59</td>
<td>0.74</td>
</tr>
<tr>
<td>O, D, SP, DP</td>
<td>933</td>
<td>7642</td>
<td>9</td>
<td>46</td>
<td>0.60</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 9. Comparison between GFADS and OC-SVM based NIDS trained with SP, DP, TF and PR features

<table>
<thead>
<tr>
<th>Features</th>
<th>Accuracy (%)</th>
<th>MR (%)</th>
<th>FAR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFADS</td>
<td>99.37</td>
<td>0.60</td>
<td>0.85</td>
</tr>
<tr>
<td>OC-SVM based NIDS [2]</td>
<td>98.29</td>
<td>4.71</td>
<td>0.0</td>
</tr>
</tbody>
</table>

80
This study shows that GFADS is suitable for high speed networks and it generates low FAR. Selecting an appropriate subset of features is also examined. Reduced number of features require less analysis time while they give similar results. GSA has a high convergence speed [7] and is confirmed by our results. PSO and GSA use agent movement to do optimization but they have some differences which make GSA more attractive. For example, only two best positions are used by PSO to calculate the direction of an agent while the GSA uses the overall force obtained by the whole agents to calculate this direction. GSA is memory-less while PSO uses memory to update velocity. As opposed to PSO, distance between solutions is considered in GSA [7]. These differences gave a better performance for GSA shown in our results. GFADS is scalable and it works with lightweight traffic. It does not include sensitive data since the lack of payload. Flow-based detection is a complement of packet-based intrusion detection. We should use both methods but in two separate stages to cover all attacks in a high speed networks. First, flow-based NIDS can detect some attacks. The packet-based method can do more analysis while some malicious activities are discovered in the first stage [5]. However, in some situations flow-based NIDS is not appropriate. For instance, in Distributed DoS attack (DDoS), for each packet passing the observation point, a flow is created. Sampling techniques are used to address this problem. GFADS also cannot detect attacks related to packet payload.

V. CONCLUSION

In this study, we proposed a GSA-based Flow Anomaly Detection System. GFADS is capable of detecting unknown attacks using ANN. We used a MLP with one hidden layer for network anomaly detection. A new optimization algorithm called GSA is deployed to optimize the interconnection weights of MLP. GFADS was evaluated using the modified version of the first public flow-based data set. The trained GFADS had 99.43% accuracy in classifying benign and malicious traffic. We compared the performance of GSA with a number of gradient descent algorithms and the PSO algorithm. In Epoch 600, the results showed the effectiveness of GFADS in detecting attacks related to packet header. The repetition of investigation with different Epochs showed that GSA converges faster than gradient descent algorithms. For cross validation, GFADS was also tested with different testing sets. The accuracy was very good in all testing sets. All results are averaged over ten experiments. Finally, we found the most efficient features subset which consists of four features. This study has been done based on centralized processing. We will expand our study in future to investigate the advantages of a distributed flow-based anomaly detection system. The time/space complexity was not considered in this study. This will also be calculated in our future study.

REFERENCES


[21] REFERENCE