Evaluating Sequential Combination of Two Light-Weight Genetic Algorithm-based Solutions to Intrusion Detection

Zorana Banković, Slobodan Bojanić and Octavio Nieto-Taladriz
ETSI Telecomunicación, Technical University of Madrid, Ciudad Universitaria s/n, 28040 Madrid
{zorana, slobodan, nieto}@die.upm.es

Abstract

In this work we have presented a genetic algorithm approach for classifying normal connections and intrusions. We have created a serial combination of two light-weight genetic algorithm-based intrusion detection systems where each of the systems exhibits certain deficiency. In this way we have managed to mitigate the deficiencies of both of them. The model was verified on KDD99 intrusion detection dataset, generating a solution competitive with the solutions reported by the state-of-the-art, while using small subset of features from the original set that contains forty one features. The most significant features were identified by deploying principal component analysis and multi expression programming. Furthermore, our system is adaptable since it permits re-training by using new data.

1. Introduction

Along with providing revolution in communication and information exchange, Internet has also provided greater opportunity for disruption and sabotage of data previously considered secure. As most of the Internet service protocols were designed at the time when Internet environment was a non-hostile one, slight attention was paid to the possibility of security flaws. The current protocols are the upgrades of the previous ones and have inherited all the security flaws which make them prone to various types of attacks. Furthermore, operating systems contain many bugs that make them susceptible to certain types of attack. The attacks to Internet service providers are carried out by exploiting these unknown weaknesses or security flaws [1].

Computer networks are usually protected against attacks by a number of access restriction policies that act as a coarse grain filter (anti-virus software, firewall, message encryption, secured network protocols, password protection). Intrusion detection systems (IDS) are the fine grain filter placed inside the protected network, looking for known or potential threats in network traffic and/or audit data recorded by hosts.

Intrusion detection systems have three common problems: speed, accuracy and adaptability. The speed problem arises from the extensive amount of data that these systems need to monitor in order to perceive the entire situation. Thus, we need to extract the most important piece of information that can be deployed for efficient detection of attacks. At this point we have used the results obtained in our previous work [2] where we deployed Principal Component Analysis (PCA) and the results obtained in [3] deploying Multi Expression Programming (MEP), in order to extract the most relevant features of the data. The features used for describing attacks are identified by deploying PCA technique, while the features used for describing normal connections are identified by MEP. In this way the total amount of data to be processed is highly reduced. As an important benefit of this arises the high speed of training the system and afterwards of its testing thus providing the possibility of real-time deployment.

Incorporation of learning algorithms provides a potential solution for the adaptation and accuracy issues of the intrusion detection problem [4]. In this work we are presenting genetic algorithm (GA) approach for classifying normal connections and intrusions. Genetic Algorithm approach is one of the forthcoming approaches in computer security and has only recently been recognized as having potential in the intrusion detection field [5], [6], mostly because of its suitability for dealing with the classification of rare classes [7].

This work represents continuation of our previous one [2] where we investigated the possibilities of applying GA to intrusion detection when only small subset of features is deployed. One of the systems was detecting only intrusions without identifying the type of the attacks, while the other one was able to identify the exact type of an attack. An important characteristic of this system is its simplicity. It is easy to understand and to train, as its training is a straightforward one. These experiments have confirmed the robustness of GA when deployed to

intrusion detection and inspired us to further continue experimenting on the subject.

Here we have further investigated a combination of two simple GA-based intrusion detection systems with the opposite qualities in the terms of detection and false-positive rate, as opposed to the existing single solutions presented by the state-of-the-art [2], [5], [6], [8]. In addition, we have used less-common serial combination of two intrusion detection systems [9], opposing to the commonly used parallel connection of multiple classification systems deployed for intrusion detection with various combinations for decision making [10]. Our aim was to mitigate the negative aspects of a certain system by supplementing another system with better performances in the terms of the same aspect.

In our serial combination, the first system exhibits very high detection rate but also high false-positive rate and the second one exhibits very low false-positive rate (lower than presented by the state-of-the-art), although lower detection rate than the first one. The combination results in significantly lower false-positive rate than the first one exhibits while maintaining high level of detection rate. In this way we have demonstrated that deploying serial connection of two GA-based systems with opposite qualities, the resulting system exhibits better characteristics than both of the original ones.

For evolving tour GA-based system KDD99Cup training and testing dataset was used [10]. KDD99Cup dataset was found to have quite drawbacks [12], [13], but despite of the shortcomings, it is still prevailing dataset used for training and testing of IDSes due to its good structure, i.e. every connection is described using 41 features and is labeled, thus providing the information whether the connection is normal or it is a specific attack type [5], [6].

In the following text Sections 2 gives the overview on GAs and IDSes and the benefits of deploying GA to intrusion detection field. Section 3 details the implementation if the system. Section 4 introduces the problem of classifying rare classes and the solutions to the problem deployed in this work. Section 5 presents the benchmark KDD99 dataset deployed for training and testing and evaluates the performance of the system on the benchmark dataset and discusses the results. Finally, the conclusions are drawn in Section 6.

2. Genetic Algorithm Approach to Intrusion Detection

Genetic algorithms (GA) are search algorithms based on the principles of natural selection and genetics. The bases of genetic algorithm approach are given by Holland [14] and it has been deployed to solve wide range of problems in computer science, engineering, economics, mathematics and many others.

The most important idea that stands beyond the initial creation of GAs is the aim of developing a system as robust and as adaptable to the environment as the natural systems are. GA operates on a population of potential solutions applying the principle of the survival of the fittest to produce better and better approximations to the solution of the problem that GA is trying to solve. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness value in the problem domain and breeding them together using the operators borrowed from the genetic process performed in nature, i.e. crossover and mutation. This process leads to the evolution of the populations of individuals that are better adapted to their environment than the individuals that they were created from, just as it happens in natural adaptation [15].

2.1. Intrusion Detection Systems – Types and Issues

According to the detection mechanism they use, exist two general categories of IDSes: misuse detection and anomaly based. Misuse detection systems are most widely used and they detect intruders with known patterns. As only the attacks that already exist in the attack database can be detected, this model needs continuous updating. Their virtue is very low false positive rate. Anomaly detection systems identify deviations from normal behaviour and alert to potential unknown or novel attacks without having any prior knowledge of them. They exhibit higher rate of false alarms, but they have the ability of detecting unknown attacks.

Another classification of IDSes is determined by the resource they monitor. According to this classification, IDSes are divided into two categories: host based and network based. Host based intrusion detection systems monitor host resources for intrusion traces whereas network based intrusion detection systems try to find intrusion signs in the network data. The current trend in intrusion detection is to combine both host based and network based information to develop hybrid systems and therefore not rely on only one methodology.

As already stated in the introduction, IDSes have three common problems: speed, accuracy and adaptability. The speed problem arises from the extensive amount of data that intrusion detection systems need to monitor in order to perceive the entire situation. In order to cope with it, the most important piece of information should be extracted so to facilitate an efficient detection of attacks. The adaptation and accuracy issues of the intrusion detection can be solved by incorporating learning algorithms. In the case of intrusion detection, learning

means discovering patterns of normal behaviour or pattern of attacks. This formulation of intrusion detection problem combines the advantages of signature-based and anomaly-based IDS. Thanks to the generalisation capability of learning algorithms, it is also possible to detect new attacks that exploit the same vulnerabilities of known attacks.

2.2. Genetic Algorithm Overview

GA evolves a population of initial individuals to a population of high quality individuals, where each individual represents a solution of the problem to be solved. Each individual is called chromosome, and is composed of a predetermined number of genes. The quality of each rule is measured by a fitness function as the quantitative representation of each rule's adaptation to a certain environment. The procedure starts from an initial population of randomly generated individuals. Then the population is evolved for a number of generations while gradually improving the qualities of the individuals in the sense of increasing the fitness value as the measure of quality. During each generation, three basic genetic operators are sequentially applied to each individual with certain probabilities, i.e. selection, crossover and mutation. Crossover consists of exchanging of the genes between two chromosomes performed in a certain way, while mutation consists of random changing of a value of a randomly chosen gene of a chromosome. Both crossover and mutation are performed with a certain possibility, called crossover/mutation rate. The algorithm flow is presented in Fig 1.

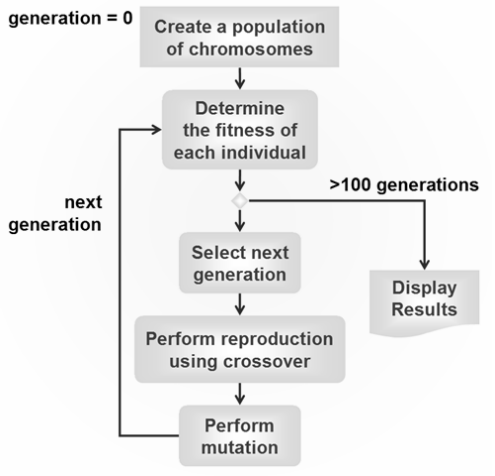

Figure1. Genetic algorithm flow

Determination of the following factors has the crucial impact on the efficiency of the algorithm: selection of fitness function, representation of individuals and the values of GA parameters (crossover and mutation rate, size of population, number of generations). This determination usually depends on the application.

2.3. The Benefits of Deploying GA to Intrusion Detection

Deployment of GA in the intrusion detection field offers number of advantages, namely:

- GAs are intrinsically parallel, since they have multiple offspring, they can explore the solution space in multiple directions at once. If one path turns out to be a dead end, they can easily eliminate it and continue working on more promising avenues, giving them a greater chance by each run of finding the optimal solution.
- Due to the parallelism that allows them to implicitly evaluate many schemas at once, GAs are particularly well-suited to solving problems where the space of all potential solutions is truly huge - too vast to search exhaustively in any reasonable amount of time, as network data is.
- Working with populations of candidate solutions rather than a single solution and employing stochastic operators to guide the search process permit GAs to cope well with attribute interactions and to avoid getting stuck in local maxima, which together make them very suitable for dealing with classifying rare class, as intrusions are.
- System based on GA can easily be re-trained, thus providing the possibility of evolving new rules for intrusion detection. This property provides the adaptability of a GA-based system, which is an imperative quality of an intrusion detection system having in mind the high rate of emerging of new attacks.

3. System Implementation

The implemented IDS is a serial combination of two IDSes. The complete system is presented in Fig. 2. The first part is a linear classifier whose false-positive rate should be reduced. As having very low false-negative rate, its decision on normal connections is considered correct. But, for its high false-positive rate, its decision on attacks is re-checked by the rule-based system. The rule-based system filters the normal connections from the potential attacks, as its rules are trained for detecting normal connections. This part of the system exhibits very low false-positive rate, i.e. the probability for an attack to be incorrectly classified as a normal connection is very low. In this way, the false-positive rate of the entire system is significantly lower than the false-positive rate of the linear classifier while exhibiting high detection rate.

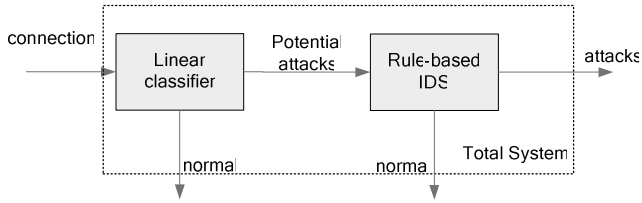

Figure 2. Block Diagram of the Complete System

As already mentioned, the first part is a simple linear classifier which classifies the connections based on the linear combination of the three features identified as those that have the highest possibility to take part in an attack by deploying PCA [2]. The three selected features and their explanations are presented in Table 1.

Table 1. The features used to describe the attacks

Name of the feature	Explication
duration	length (number of seconds) of the connection
src_bytes	number of data bytes from source to destination
dst_host_srv_serror_rate	percentage of connections that have "SYN" errors

The linear classifier is evolved using GA algorithm as described in the previous section. Each chromosome, i.e. potential solution to the problem, in the population is comprised of four genes, where the first three represent coefficients of the linear classifier and the fourth one represents the threshold value. The decision whether the current connection is an attack is made according to the formula (1):

$$\text{gene}(1)*\text{con}(\textit{duration})+\text{gene}(2)*\text{con}(\textit{src\_bytes})+\text{gene}(3)*\text{con}(\textit{dst\_host\_srv\_serror\_rate})<\text{gene}(4) \quad (1)$$

where con(*duration*), con(*src_bytes*) and con(*dst_host_srv_serror_rate*) are the values of the *duration, src_bytes* and *dst_host_srv_serror_rate* feature of the current connection.

The linear classifier was trained using incremental, i.e. the algorithm where the number of individuals is increasing in every generation and a certain number of the worst individuals is substituted in each generation with the newly-bred ones. The population contained 1000 individuals which were trained during 300 generations. The mutation rate was 0.1 while the crossover rate was 0.9. The previous numbers were chosen after certain number of experiments. The size of the population and the number of generations are selected in the manner that their further increasing doesn't bring significant performance improvement nor overfitting. The type of crossover deployed was uniform crossover, i.e. a new individual had equal chances to contain either of the genes of both of its parents. The performance measurement, i.e. the fitness function, was the squared percentage of the correctly classified connections, i.e. according to the formula:

$$fitness = \left(\frac{count}{numOfCon} \right)^2 \quad (2)$$

where *count* is the number of correctly classified connections, while *numOfCon* is the number of connections in the training dataset. The squared percentage rather than the simple percentage value was chosen because it exhibited better. The result of this GA was its best individual which forms the first part of the system presented in Fig.2.

The second part of the system presented in Fig. 2 is a rule-based system, where simple *if-then* rules for distinguishing normal connections are evolved. For that reason, the most important features for a normal connection are identified using Multi Expression Programming [3]. These three features and their explanations are listed in Table 2.

Table 2. The features used to describe normal connections

Name of the feature	Explication
service	Destination service (e.g. telnet, ftp)
hot	number of hot indicators
logged in	1 if successfully logged in; 0 otherwise

An example of a rule can be the following one:

if *(service="http"* ***and*** *hot="0"* ***and*** *logged_in="0")*
then *normal;*

The rules were trained using incremental GA, 500 individuals that were trained during 300 generations, with crossover and mutation rate 0.9 and 0.1 respectively. The selection of the number of generations and the population size is performed analogously to the linear classifier. In this case simple one–point crossover was used because the change of the crossover type doesn't make a difference in this case. The result of the training was a set of 200 best-performed rules. The performance measurement (the fitness function) in this case was the F-value with the parameter 0.8:

$$fitness = \frac{1.8 * recall * precision}{0.8 * precision + recall}$$
$$precision = \frac{TP}{TP + FP} \quad (3)$$
$$recall = \frac{TP}{TP + FN}$$

where *TP*, *FP* and *FN* make parts of a confusion matrix typically used to evaluate performance of a machine learning algorithm presented in Table 3. In classification problems, class "C" is the class of the interest, i.e. that class that is being detecting (in this case normal connections) and "NC" as a conjunction of all the other classes. Parameter value of 0.8 was chosen after

performing few experiments with different values as the value that provided the best performances of the system in the terms of both detection and false-positive rate.

Table 3. Confusion matrix for defining four possible outcomes when classifying class "C"

	Predicted Class "C"	**Predicted Class "NC"**
Actual Class "C"	True Positives (TP)	False Negative (FN)
Actual Class "NC"	False Positives (FP)	True Negative (TN)

The algorithm was performed as presented in Section 2. The system presented here was implemented in C++ programming language. The software for this work used the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology [16]. The time of training the implemented system is 185 seconds while the testing process takes 45 seconds. The system was demonstrated on AMD Athlon 64 X2 Dual Core Processor 3800+ with 1GB RAM memory on its disposal.

4. Imbalanced Classes Problem in Intrusion Detection

The task of detecting intrusions belongs to the problem of detecting so-called rare or imbalanced classes since intrusions occur rarely, i.e. in real-world the percentage of intrusive data is very small comparing to the percentage of normal data. Besides, some intrusions occur more rarely than the others, which make the classification task even more complicated. Conventional learning techniques exhibit certain deficiencies when dealing with rare classes [7]. The most important shortcoming is the tendency for generalization, which usually results in wrong classification of the instances of rare classes.

As stated before, genetic algorithms are global search techniques that work with populations of candidate solutions rather than a single solution and employ stochastic operators to guide the search process. These characteristics permit genetic algorithms to cope well with attribute interactions and avoid getting stuck in local maxima. In this way both generalization and data fragmentation are avoided which are both inappropriate for dealing with rare classes. Besides this intrinsic capability of GAs, we have deployed F-measure as the evaluation metrics that is proven to be very suitable when dealing with imbalanced classes [7]. F-measure is a combination of precision and recall. The precision of a classification rule, or set of rules, is the percentage of times the predictions associated with the rule(s) are correct. If these rules predict class *X* then recall is the percentage of all examples belonging to *X* that are covered by these rule(s). Rare cases and classes are valued when using these metrics because both precision and recall are defined with respect to the positive (rare) class.

5. Results

Learning algorithms have a training phase where they mathematically 'learn' the patterns in the input dataset. The input dataset is also called the training set which should contain sufficient and representative instances of the patterns being discovered. A dataset instance is composed of features, which describe the dataset instance. Learned patterns can be used to make predictions on a new dataset instance based on its diversity from normal patterns or its similarity to known attack patterns or a combination of both.

In order to promote the comparison of advanced research in the area of intrusion detection, the Lincoln Laboratory at MIT, under DARPA sponsorship, conducted the 1998 and 1999 evaluation of intrusion detection [17]. Based on binary TCP dump data provided by DARPA evaluation, millions of connection statistics are collected and generated to form the training and test data in the Classifier Learning Contest organized in conjunction with the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 1999 (KDD-99) [11]. The learning task was to build a detector (i.e. a classifier) capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" or normal connections.

5.1. Training and Testing Datasets

The dataset contains 5,000,000 network connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows from a source IP address to a target IP address under some well defined protocol [18]. The training portion of the dataset (labelled as "kdd_10_percent") contains 494,021 connections of which 20% are normal. Each connection record contains 41 independent fields and a label (normal or type of attack). Each attack belongs to one of the four attack categories: user to root, remote to local, probe, and denial of service. The testing dataset (labelled "corrected") provides a dataset with a significantly different statistical distribution than the training dataset and contains an additional 14 (unseen) attacks not included in the training dataset. The basic characteristics of the datasets are given in Table 4.

Table 4. The basic characteristics of KDD99 datasets

Dataset Label	Number of Attacks	Number of Normal Connections
kdd_10_percent	396743	97277
corrected	250436	60593

5.2. Dataset Issues

The most important flaws of the mentioned dataset are the following ones [12]:

- The dataset contains biases that may be reflected in the performance of the evaluated systems.
- None of the sources explaining the dataset contains any discussion of the data rate, and its variation with time is not specified.
- The skewed nature of the attack distribution may represent a bias that affects the results of the evaluation.
- There is no discussion of whether the quantity of data presented is sufficient to train a statistical anomaly system or other learning-based system.

Furthermore, in [13] is demonstrated that the transformation model used for transforming raw DARPA's network data to a well-featured data item set is 'poor'. Here 'poor' refers to the fact that some attribute values are the same in different data items that have different class labels. Due to this, some of the attacks can't be classified correctly.

5.3. Obtained Rates

The system was trained using "kdd_10_percent" and tested on "corrected" dataset. The obtained results are summarized in Table 5. Presented rates of the linear classifier and whole system are the rates for detecting attacks, while the rates of the rule-based system are for detecting normal connections. The last column gives the value of classical F-measure so that learning results could be easily compared with a unique feature for both recall and precision. Our previous statement of high reducing of the false-positive rate while maintaining high detection rate is confirmed, as the false-positive rate is reduced from 40.7% to 2.7%, while the detection rate has reduced for only 0.15%. The increasing of F-value is also exhibited.

The adaptability of the system was tested as well by first training the system with a subset of "kdd_10_percent" (250,000 connections out of 491,021). The generated rules were taken as the initial generation and re-trained with the remaining data of "kdd_10_percent" dataset. Both of the systems were tested on "corrected" dataset. The system exhibited improvements in both detection and false positive rate. The improvements are presented in the Table 6.

Table 5. The performances of the whole system and its parts separately

System	Detection rate		False Positive Rate		F-measure
	Num.	Per. (%)	Num.	Per. (%)	
Linear Classifier	231030	92.25	24628	40.7	0.913
Rule-based	45504	75.1	5537	2.2	0.815
Whole system	230625	92.1	862	1.4	0.96

5.4. Discussion

The final results are similar to those presented in [5] and [6], although we have used smaller subset of features. Hence, our system can perform the training process and the process of detecting intrusions faster while maintaining high detection rates.

Table 6. The performance of the system after re-training

System	Detection rate		False Positive Rate		F-measure
	Num.	Per (%)	Num	Per (%)	
Whole system after trained with a subset of "kdd_10_percent"	183060	73.1	1468	2.4	0.84
Whole system after re-trained with the rest of the data from "kdd_10_percent"	231039	92.3	862	1.4	0.96

The drawbacks of the dataset have influenced the gained rates. As reported in [13], some of the newly introduced attacks from the testing dataset are very similar to the normal connections which make them very prone to incorrect classification. As comparison, the detection rate of the system tested on the same data that it was trained on, i.e. "kdd_10_percent", is 99.2% comparing to the detection rate of 92.1% after testing the system using "corrected" dataset. The decreasing of detection rate by 8% is obtained due to the significantly different statistical distribution of the datasets. In addition, the distribution of the attacks and normal connection in the datasets is not very realistic [12], i.e. only 20% of the training data set makes normal connections while in real world the situation is quite opposite, as the percentage of normal packets highly exceeds the percentage of intrusive ones. This distribution is highly inconvenient for training anomaly systems (as this system is). Thus, everything

stated here had negative effect on the rates obtained in this work.

The adaptability of the system was also tested by training the system first with a fraction of "kdd_10_percent" and after that training the obtained system with the rest of the dataset. Improvements in both detection and false-positive rates were achieved as presented in Table 6. Thus, it is demonstrated that the system is adaptive since it exhibited improvements after being trained with new data.

6. Conclusions

In this work a serial combination of two GA-based IDSes with opposite qualities is introduced. The properties including adaptability of the resulting system were analyzed. The proposed combination is demonstrated to be very favorable for mitigating the negative aspects of the first system in the series. As our system uses only six features to describe the data, its time of training and decision making is considerably reduced, thus providing the possibility of real-world deployment.

As previously stated in the Introduction, three common problems of intrusion detection systems are speed, accuracy and adaptability. In this work, the problem of the speed in addressed by deploying small subsets of features for describing network connections. In this way, the periods of training and testing a certain system have been highly decreased. Introducing incremental genetic algorithm as the approach for evolving the population has also very positive impact on the time of training since the populations contains small number of individuals at the beginning of the process of evolution. Next, the gained performances of the presented systems demonstrate high accuracy of the implemented system. Finally, adaptability of the implemented system has been tested by re-training the systems with additional data. After the process of re-training, enhancement of performances has been confirmed. Thus, it is demonstrated that the system implemented in this work has successfully addressed and solved the problems of intrusion detection systems.

The benefits of deploying GAs to intrusion detection have also been demonstrated. Due to their possibility of fast searching of the space of the possible solutions high detection rate was achieved within small amount of time. The possibility of re-training the results obtained after a process of evolution has resulted in high adaptability of the system to the changes of environment. Unfortunately, due to the dataset used for training and testing whose distribution of data is not very realistic (only 20% of data are normal connections) the possibility GAs of detecting rare events couldn't be demonstrated. Part of our future work will be dedicated to the proper adjustment of the dataset, since simple over-sampling and under-sampling are reported to exhibit weaknesses [7] resulting in degraded performances of the trained system.

As real-world network data is unlabeled, and considering that labeling network data would be an extensive engineering task, the algorithm could be adapted to be able to work with unlabeled data. This can be performed only by defining appropriate fitness function. The fitness function should be able to properly define the performance of the individual without previously knowing whether a connection is an attack or not.

The principal idea of this work was to indicate the advantages of deploying such a system in intrusion detection. Our future work will consist in pursuing a real application of the system presented here, thus we will be able to provide the results based on real-world network data.

6. Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science under project TEC2006-13067-C03-03 and also by the European Community Research Programme under the FastMatch project which is partially supported by the European Community under the Information Society Technologies (IST) priority of the 6th Framework Programme for R&D (IST—27095, www.fastmatch.org).

7. References

[1] McHugh, J., Christie, A., Allen, J.: Defending Yourself: The Role of Intrusion Detection Systems, IEEE Software, Sept./Oct. 2000, pp. 42-51

[2] Z. Banković, D. Stepanović, S. Bojanić, and O. Nieto-Taladriz, "Improving Network Security Using Genetic Algorithm Approach", to be published in Computers & Electrical Engineering, Elsevier

[3] C. Grosan, A. Abraham, and M. Chis, "Computational Intelligence for light weight intrusion detection systems", International Conference on Applied Computing (IADIS'06), San Sebastian, Spain, ISBN: 9728924097, 2006. pp. 538-542,

[4] C.Elkan, "Results of the KDD'99 Classifier Learning", ACM SIGKDD Explorations, 1, 2000, 63-64.

[5] R. H Gong, M. Zulkernine, P. Abolmaesumi, "A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection", Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN`05), 2005.

[6] A. Chittur, "Model Generation for an Intrusion Detection System Using Genetic Algorithms", http:// www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf, accessed in 2006.

[7] G. Weiss,"Mining with rarity: A unifying framework". SIGKDD Explorations 6(1):7–19. 2004

[8] W. Lu, I. Traore, "Detecting New Forms of Network Intrusion Using Genetic Programming", Computational Intelligence, Volume 20, Number 3, pp. 470-490, 2004.

[9] E. Tombini, H. Debar, L. Me, M. Ducasse, "A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP traffic", Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)

[10] G. Giacinto, F. Roli and L. Didaci, Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recog. Lett.* 24 12 (2003), pp. 1795–1803.

[11] KDD Cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, October 1999

[12] J .McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA IDS Evaluation as Performed by Lincoln Laboratory", ACM Trans. on Information and System security, Vol. 3, No.4, Pages 262-294, November 2000.

[13] Y. Bouzida and F. Cuppens, "Detecting known and novel network intrusion", IFIP/SEC 2006 21st IFIP TC-11 International Information Security Conference Karlstad University, Karlstad, Sweden. May 2006.

[14] J. Holland, *Adaptation in natural and artificial system*, Ann Arbor, The University of Michigan Press (1975)

[15] D. E Goldberg, *Genetic algorithms for search, optimization, and machine learning*. Addison-Wesley (1989)

[16] GAlib A C++ Library of Genetic Algorithm Components, http://lancet.mit.edu/ga/

[17] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K Cunningham., M.A. Zissman, "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation", Proceedings of the 2000 DARPA Information Survivability Conference and Exposition, 2 (2000)

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