Efficient Rule Generation for Cost-Sensitive Misuse Detection Using Genetic Algorithms*

Saqib Ashfaq
Lahore University of Management Sciences
Lahore, Pakistan
sashfaq@lums.edu.pk

M. Umar Farooq
Lahore University of Management Sciences
Lahore, Pakistan
05030002@lums.edu.pk

Asim Karim
Lahore University of Management Sciences
Lahore, Pakistan
akarim@lums.edu.pk

Abstract

This paper presents a genetic algorithm (GA) for generating efficient rules for cost-sensitive misuse detection in intrusion detection systems. The GA employs only the five most relevant features for each attack category for rule generation. Furthermore, it incorporates the different costs of misclassifying attacks in its fitness function to yield rules that are cost sensitive. The generated rules signal an attack as well as its category. The GA is implemented and evaluated on the KDDCup 99 dataset. Its detection performance is comparable to the winners of the KDDCup 99 competition. However, the rules generated by our GA are short and amenable for real time misuse detection.

1. Introduction

Computer networks around the world are facing intense security challenges posed by the increasing frequency and sophistication of malicious attacks. Tackling these attacks is a major problem in the computer science community. In recent years, computational intelligence approaches to network intrusion detection have shown promise with significant potential for the future [1].

The majority of intrusion detection systems (IDS) today employ misuse detection to identify previously known attacks. Misuse detection requires a prior knowledge of attack characteristics (or signatures) and typically works by pattern matching usage behaviors with the known signatures. As such, misuse detection cannot identify new attacks for which signatures have not been developed. Moreover, the signature database is usually updated manually which is tedious and error prone. Many IDSs including Snort, a popular open-source IDS, implement signatures as rules. Therefore, there is a need to automate the process of rule generation for misuse detection from attack databases.

Genetic algorithm (GA) is a computational intelligence technique that provides guided random search of large search spaces. The problem of generating rules from attack databases can be posed as a search problem where the search space is the space of all possible rules and the desired rules are the ones that characterize attacks with a high degree of accuracy.

In this paper, we present an efficient rule generator for misuse detection using a genetic algorithm. The idea of using genetic algorithm for rule generation is not new. However, many previous rule generation approaches for misuse detection utilize features of the usage behaviors as input without regard to their relevance. We develop a GA that uses 5 most relevant features identified in [2] and produces results that are comparable to the winners of the KDDCup 99 competition. As such, the rules generated by our algorithm are short and amenable to real time misuse detection.

2. Related work

Genetic algorithms (GAs) provide an effective approach for finding acceptable classification rules from a large space of potential rules [3, 4]. Recently, GA and other evolutionary computing approaches have received significant attention for the design and implementation of intrusion detection systems [1]. In [5], a hybrid fuzzy-genetic approach is adopted for an intelligent IDS. However, in their approach the GA is used to tune the membership functions of fuzzy sets, and is not involved in the generation of signatures or rules for misuse detection. Pillai et al. apply GA for misuse detection and rule generation, but their GA formulation does not incorporate the crossover

* This work was supported by a PTCL R&D grant, LUMS graduate fellowship (Ashfaq), and HEC graduate scholarship (Farooq)

operation which is an important operation in evolutionary computation [6]. The genetic programming (GP) approach is used for misuse detection in [7]. They optimize their approach to efficiently utilize the memory hierarchy in computers. Eight basic attributes of KDDCup 99 dataset are used for rule generation and the classification results obtained are comparable to the winners of the KDDCup 99 competition [8, 9].

An important issue in rule generation for misuse detection is identification of significant features. By using a significant feature set, not only will the rules so generated be of higher quality but they will also make the pattern matching procedure more efficient. This latter advantage is essential for practical IDSs that need to operate in real time by matching usage behaviors with rules on the fly. Middlemiss and Dick perform weighted feature extraction for misuse detection using GA [2]. They determine the most relevant features for each category of attack from KDDCup 99 dataset. This feature set represents a reduced set and its usage is likely to produce effective and efficient rules for misuse detection.

3. Genetic algorithm for cost-sensitive misuse detection

We present a genetic algorithm for misuse detection using the five most relevant features identified in [2]. The algorithm generates "if…then" rules that identifies an attack as well as its category so that appropriate action can be taken in response. Furthermore, our misuse detection approach is cost sensitive that considers the cost of false alarms for each category of attack separately. We use the Michigan approach for rule representation where one rule is represented by a single chromosome (an individual in the population) [3]. Moreover, rules are determined for each attack category separately. The detailed description of our algorithm is presented in the following subsections.

3.1. Dataset and relevant feature selection

We use the KDDCup 99 competition dataset for implementing and evaluating our algorithm [8]. This dataset is prepared by DARPA intrusion detection evaluation program and is commonly used in the literature for evaluating the performance of intrusion detection systems. This dataset represents the real world in which the algorithm is expected to operate. Furthermore, it is a rich dataset containing millions of records described by 41 categorical and numeric attributes. The attributes are categorized as 9 basic (e.g. duration, protocol_type) and 32 derived attributes (e.g. logged_in, is_guest_login).

Table 1. IDs of the top five ranked features for each attack category in KDDCup 99 dataset

Rank	DoS	R2L	U2R	Probe
1	23	33	33	2
2	34	3	6	37
3	1	12	31	35
4	11	32	41	3
5	24	36	17	6

Table 2. Cost matrix for misuse detection

	Normal	Probe	DoS	U2R	R2L
Normal	0	1	2	2	2
Probe	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

The dataset contain instances of several attacks including smurf, ipsweep, imap, 4_write, nmpa, neptune, and satan. These attacks are grouped into four attack categories: DoS (denial-of-service), probe, U2R (user-to-root), and R2L (remote-to-local). Our algorithm is capable of identifying the category of the attack.

Many existing algorithms use 9 basic features for attack classification. We use the five most relevant attributes from among the 41 attributes for each attack category. These are the top five ranked attributes found by [2] as shown in Table 1. The numbers in the table represent the attribute ID in the dataset which ranges from 1 to 41. The most relevant attributes are both categorical and numeric. Each of the numeric attribute is converted to a set of binary attributes using equi-width binning. The motivation for using a reduced set of attributes is as follows: (1) to reduce the computation time for rule generation using GA, and (2) to enable efficient real-time pattern matching of rules with usage behaviors.

3.2. Cost matrix for misuse detection

It is known that the cost of misclassifying an attack is different for different attacks. Our algorithm is capable of using a cost matrix which associates a cost with the misclassification of attack category i as attack category j for all i and j. For the implementation in this work, we use the cost matrix given in Table 2 [9]. In this table, the columns identify the attack categories detected while the rows identify the actual attack categories.

3.3. Representation

A fundamental step in the design of a GA is the compatible representation of the problem. For the rule generation problem, two representations have been used in the literature: Michigan and Pittsburgh [3]. In the Michigan approach, each individual in the population (a chromosome) encodes a single rule whereas in the Pittsburgh approach each chromosome encodes a set of rules. Both approaches have their pros and cons. We adopt the Michigan approach in this work because of its efficiency resulting from the smaller individuals generated by the approach.

A rule consists of two parts: the antecedent and the consequent. In general, the antecedent part is a conjunction of a number of attribute values. Each categorical attribute in the conjunction is represented by k bits, where k is the number of possible values for the attribute. If a particular value is present then the corresponding bit is set to 1; otherwise, it is zero. Multiple bits may be set to 1 representing a logical OR among the corresponding attribute values. Multiple attributes in the antecedent are chained by the logical AND operator. Numeric attributes are represented by k bits, where k is the number of bins into which the attribute is categorized using equi-width binning. If the value of the attribute lies within a bin, then the corresponding bit is set to 1; otherwise, it is zero.

The consequent part consists of the attack category attribute and its value only. It is represented by m bits, where m is the number of attack categories. If an attack category is present then the corresponding bit is set to 1 and other bits are zero. The consequent part is encoded in the genome of the individual and participates in the evolution process. Our algorithm is run m times, once for each possible value (attack category), generating the best rules for each attack category.

3.4. Fitness function

The fitness of each rule (an individual or chromosome) is calculated separately based on the cost matrix presented in Table 2. Mathematically, the fitness value of a rule i is given by

$$F_i = -\sum_j N_{ij} C_{jk}$$

where N_{ij} is the number of instances (records) in the dataset that satisfy the antecedent part of rule i and have attack category j, and C_{jk} is the cost associated with attack category j misclassified as category k by rule i (from Table 2). The larger the fitness value, the better the rule for classifying the attack category.

3.5. Selection, crossover, and mutation

The initial selection for the rules (or individuals in the population) can be done randomly or by using the training dataset to select rules that cover many records. In subsequent iterations of the GA, selection is done based on the fitness value of the rules.

The crossover operation is an important step in a GA. One-point and two-point crossover operations are the most commonly used types. In the one-point crossover operation, the two rules taking part in the operation are cut at a randomly selected point and their left (or right) parts are interchanged. This crossover operation is sensitive to the position of the attributes within the chromosome. For example, the left most and right most attributes are highly unlikely to go in the same partition. We use the two-point crossover operation, in which two cut points are randomly selected in the two rules and the part within the cut point is interchanged. Two-point crossover is independent of the position of the attributes in the rule. The selection of chromosomes that take part in the crossover operation is done by the roulette wheel strategy. The population size is kept constant from one iteration to the next.

Mutation is random flipping of the bits in the rule. It helps avoid getting stuck in local maxima in the search. We use single bit mutation with a low flipping probability.

4. Implementation and Evaluation

The genetic algorithm for cost-sensitive misuse detection is implemented in Java and tested and evaluated on the KDDCup 99 dataset [8]. We use the 10% labeled data (file name: kddcup.data_10_percent.gz) for training and the corrected data (file name: corrected.gz) for evaluation of the algorithm. The GA is run 4 times, once for each attack category, to obtain the best rule for classifying each attack category. In general, the results of our GA are comparable to those of the winning entry of the KDDCup 99 competition [9]. However, the rules generated by our algorithm are shorter and thus more efficient for real time applicability.

The performance of our GA for DoS attack classification on the training and corrected datasets is given in Table 3. These results are obtained after 5 iterations of the GA. It is observed that the percentage correct classification increases drastically when the

Table 3. Performance results for DoS attack after 5 iterations

No. of Rules	Fitness Value		Percentage Correct	
	Training Dataset	Corrected Dataset	Training Dataset	Corrected Dataset
10	-391458	-223298	20.76	28.20
150	-90243	-61738	71.73	80.15
200	-49629	-53633	89.95	82.77
500	-49604	-52999	89.995	82.96

Table 4. Performance results for all attacks after 50 iterations and using 200 rules

	Percentage Correct			
	Probe	DoS	U2R	R2L
Training Dataset	97.55	71.73	97.88	97.79
Corrected Dataset	97.66	80.15	97.99	96.44

number of rules (the population size) is increased from 10 to 150. Increasing the population size beyond 200 does not increase the correct classification significantly. The fitness values of the best rules are also given in the table.

Based on the above result for DoS attacks, two hundred rules appear to be an adequate population size. Using this population size of 200, the performance of the GA after 50 iterations and for all attack categories is given in Table 4. It is seen that the performance does not increase significantly by increasing the number of iterations. The performance of our GA is compared to that of the KDDCup 99 winner in Figure 1. Our GA outperforms the winning result for probe and U2R, and is not significantly worse for R2L and DoS attack categories.

These results highlight the advantage of using a reduced and relevant feature set. The performance is comparable to the winning performance and at the same time being more efficient.

5. Conclusion

The majority of the algorithms presented in the literature for misuse detection use the 9 basic features of traffic flows as input. We present an efficient rule generator for misuse detection that uses the five most relevant features for each attack category. This reduced and relevant input speeds up the rule generation process and enables real time operation of misuse detection systems. We present a genetic algorithm (GA) for the rule generation. The generated rules are cost sensitive and are capable of identifying and classifying attack categories. The classification results

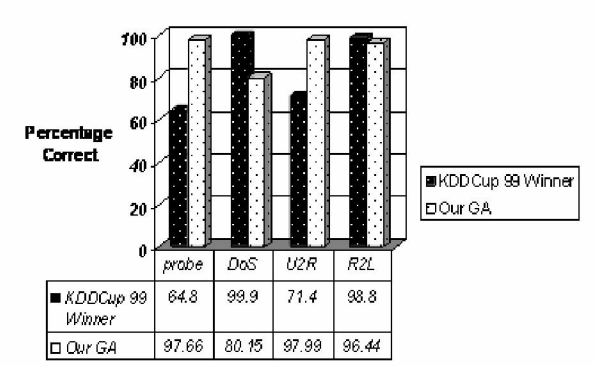

Figure 1. Performance comparison of our GA with KDDCup 99 winner

of our GA generated rules are comparable to those of the winners of the KDDCup 99 competition.

5. References

[1] A. Karim, "Computational Intelligence for Network Intrusion Detection: Recent Contributions," *International Conf. on Computational Intelligence and Security (CIS 05)*, LNAI 3801, Springer, 2005, pp. 170-175.

[2] M.J. Middlemiss, G. Dick, "Weighted Feature Extraction Using a Genetic Algorithm for Intrusion Detection," *2003 Congress on Evolutionary Computation (CEC-03)*, 2003, pp. 1669-1675.

[3] A.A. Freitas, "A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery," In A. Ghosh and S. Tsutsui (eds.), *Advances in Evolutionary Computation*. Springer, 2002, pp. 819-845.

[4] M.V. Fidelis, H.S. Lopes, A.A. Freitas, "Discovering Comprehensible Classification Rules with a Genetic Algorithm," *2000 Congress on Evolutionary Computation (CEC-00)*, 2000, pp. 805-810.

[5] S.M. Bridges, R.M. Vaughn, "Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection," *National Information Systems Security Conference (NISSC)*, 2002.

[6] M.M. Pillai, J.H.P. Eloff, H.S. Venter, "An Approach to Implement a Network Intrusion Detection System Using Genetic Algorithms," *SAICSIT 2004*, 2004, pp. 221–228.

[7] D. Song, M.I. Heywood, A.N. Zincir-Heywood, "Training Genetic Programming on Half a Million Patterns: an Example from Anomaly Detection," *IEEE Transactions on Evolutionary Computation*, 9(3), 2005, pp. 225-239.

[8] UCI Machine Learning Repository, "KDDCup'99 Data." http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[9] C. Elkan, "Results of the KDD'99 Classifier Learning Contest," http://www.cse.ucsd.edu/users/elkan/clresults.html, 1999.