INTRUSION DETECTION USING SURROGATE MODEL ON TAG3P

[1]VAN CANH VU, [2]TUAN HAO HOANG

[1,2]Le Quy Don technical University, Ha Noi, Vietnam
E-mail: canhvuvan@yahoo.com, haohth@gmail.com

Abstract— Nowadays, hackers use different types of attacks for getting the valuable information. Many intrusion detection techniques, methods and algorithms help to detect these attacks. In this paper, a rule evolution approach using Surrogate model on Tree Adjoining Grammar Guided Genetic Programming (SuTAG3P) for generating new rules. A testing dataset proposed by DARPA is used to evaluate these new rules. The proof of concept implementation shows that a rule generated by SuTAG3P has a low false positive rate (FPR), a low false negative rate and a high true positive rate of detecting attacks.

Index Terms— attack detection; intrusion detection system; tag3p, genetic programming, surrogate model.

I. INTRODUCTION

Nowadays, computer network systems play an increasingly important role in our society and economy. They have become the targets of a wide array of malicious attacks that invariably turn into actual intrusions. This is the reason computer security has become an essential concern for network administrators. Intrusion detection systems are designed to monitor a network or systems for malicious activities or policy [6][7][9][36] occurring in a computer system or network and analyse them for signs of possible incidents, which are violations or forthcoming threats of violation of computer security policies, acceptable utilized policies, or standard security practices. Intrusion incidents to computer systems are increasing because of the commercialization of the internet and local networks [37] and new automated hacking tools. Computer systems are turning out to be more and more susceptible to attack, due to its extended network connectivity.

Intrusion detection has been extensively studied since the seminal report written by Anderson [1]. Normally, intrusion detection techniques are classified by detection approaches: misuse detection and anomaly detection. Signature-based or misuse detection techniques monitor packets and compare with pre-configured and pre-determined attack patterns, which can be described by specific patterns or sequences of events and data. On the other hand, statistical anomaly detection techniques model system or users' normal behaviors, and any deviation from the normal behaviors is considered as an intrusion. Misuse detection techniques have low false detection rates (FDR), but their major weakness is that novel or unknown attacks will go unnoticed until corresponding signatures are added to the database of the Intrusion Detection System (IDS). Anomaly detection techniques have the potential to detect unknown attacks, but quite often they tend to have high false detection rates because it is very difficult to discriminate between abnormal and intrusive behavior.

In this paper we propose the rule evolution approach using Surrogate with Tree Adjoining Grammar Guided Genetic Programming (SuTAG3P) to detect novel attacks on network. Initial rules are selected from known attacks based on background knowledge in DARPA dataset [26]. These initial rules can be represented as derived trees in TAG3P, which will be evolved to generate new rules are used to detect novel or known attacks. To evolve and evaluate these new rules, we use the training and testing dataset proposed by DARPA [26], which includes almost all known network based attacks namely land, synflood, ping of death (pod), smurf ieardrop, back, Neptune, Ipsweep, portsweep and UDPstorni attacks. The proof of concept implementation shows that the TAG3P based approach can detect smurf and UDPstorm attacks, which are absent from the training dataset.

The remainder of the paper is organized as follows. Section 2 presents an overview of related works. Section 3 provides background knowledge on Genetic Programming, Tree Adjoining Grammar Guided Genetic Programming and using Surrogate model on TAG3P. Section 4 introduces how to use SuTAG3P to generate new rules and discusses the experimental results. Finally, section 5 makes some concluding remarks.

II. RELATED WORKS

Laskov et al. [22] develop an experimental framework for comparative analysis of supervised (classification) and unsupervised learning (clustering) techniques for detecting malicious activities. They define two scenarios for evaluating the aforementioned learning algorithms from both categories. In the first scenario, they assume that training and test data come from the same unknown distribution. In the second scenario, they consider the case where the test data comes from new (i.e., unseen) attack patterns. The results of [22] show that the supervised algorithms in general show better classification accuracy on the data with known attacks. Among these algorithms, the decision tree algorithm has achieved the best results (95% true positive rate, 1% false-positive rate). However, if there

are unseen attacks in the test data, then the detection rate of supervised methods decreases significantly.

Lee and Solfo [23] build a classifier to detect anomalies in networks using data mining techniques. They implement two general data mining algorithms that are essential in describing normal behavior of a program or user. They propose an agent-based architecture for intrusion detection systems, where the learning agents continuously compute and provide the updated detection models to the agents.

Sommer and Paxson [34] study the imbalance between the extensive amount of research on ML-based intrusion detection and the lack of operational deployments of such systems. They identified challenges particular to network intrusion detection and provide a set of guidelines for fortifying future research on ML-based intrusion detection.

Pham et al. [29] used Machne Learning techniques for web intrusion detection, their experiments compared some ML tecniques and showed that Logistic regression is the best result for web intrusion detection.

Sinclair et al. [33] use genetic algorithms and decision trees to create rules for an intrusion detection expert system, which supports the analyst's job in differentiating anomalous network activity from normal network traffic. In this work, GA is used to evolve simple rules for network traffic. Each rule is represented by a genome and the initial population of genomes is a set of random rules. Each genome is comprised of 29 genes: 8 for source IP, 8 for destination IP, 6 for source port, 6 for destination port, and 1 for protocol.

Li [24] describes a few disadvantages of the algorithm proposed in [33] and defined a new technique for defining IDS rules. They argue that in order to detect intrusive behaviors for a local network, network connections should be used to define normal and abnormal behaviors.

Mukkamala et al. [27] describe approaches to intrusion detection using neural networks and Support Vector Machines. Their goal is to discover patterns or features that describe user behavior to build classifiers for recognizing anomalies.

Gomez and Dasgupta [5] show that with fuzzy logic, the false alarm rate in determining intrusive activities can be reduced. They define a set of fuzzy rules to define the normal and abnormal behavior in a computer network, and a fuzzy inference engine to determine intrusions. They use a genetic algorithm to generate fuzzy classifiers, which is a set of fuzzy rules in the form defined above.

Hofmeyr and Forrest [11] proposed the first immune-inspired model applicable to various computer security problems, which is specialized to detect intrusions in local area networks based on TCP/IP. Kim et al. [17] provide an introduction and analysis of the key developments within the field of immune-inspired computer security as well as suggestions for future research. They summarize six immune features that are desirable for an effective IDS: distributed, multi-layered, self-organised, lightweight, diverse and disposable. Zamani et al. [38][39] describe an artificial immune algorithm for intrusion detection in distributed systems based on danger theory, an immunological model based on the idea that the immune system does not recognize between self and non-self, but rather between events that cause damage.

Surrogate models can be used to support EAs in several ways. Most of research uses the surrogate model to evaluate individuals, and this information can also be used to preselect the most promising individuals that are then evaluated with the expensive fitness function [12]. That way, individuals classified as poor by the surrogate model can be quickly discarded, while the better individuals are fully evaluated to allow an accurate ranking. Furthermore, the full evaluations of these individuals are used to further refine the surrogate model particularly in the most promising areas. Others have proposed to alternate between generations that are evaluated with the surrogate model and generations that are evaluated using the expensive fitness function [12][31], to use surrogate models for supporting local search [25], or for pre-selection among individuals generated by a genetic operator [3].

However, so far, successful use of surrogates to speed up EAs has been mostly restricted to fixed-length representations for which it was easy to define a distance metric. GP, on the other hand, usually uses a tree representation of variable length. Machine learning or statistical techniques cannot be used directly to work with such genotypes or phenotypes, which probably is the reason why, to the best of our knowledge, only recently work on using surrogate functions with GP has been reported [18]. As an alternative to their approach, we overcome the difficulties of applying surrogate models with GP by using an (approximate) phenotypic characterization of the individual, instead of working on a syntactic level. As it is a vector consisting of integer values, we can then apply standard techniques from the surrogate literature to estimate the fitness value of a GP individual without performing an expensive fitness evaluation.

As indicated in [8], there is a positive correlation between diversity and performance achieved by GP for many problems, and consequently high diversity is considered beneficial. A typical approach to maintaining diversity is fitness sharing, it forces similar individuals to share their fitness values and thereby discourages the formation of clusters of very similar individuals. Recent research by Su Nguyen [28] aims at incorporating sampling semantics to form clusters sharing their fitnesses, the sampling semantics is in some respects similar to our phenotypic characterization. The amount of duplicates can be reduced to a large extent without additional simulations, however. Branke [2] used a simplified

version of the phenotypic characterization to cheaply compute approximate phenotypic equivalence between trees and could demonstrate its usefulness to improve convergence speed and solution quality of standard GP. Here we further analyze this approach and extend it to create surrogate functions to improve the convergence of GP rewrite it.

In the following section, we will propose a method for using surrogate model in TAG3P to evolve new intrusion detection rulers.

Review Stage

Submit your manuscript electronically for review.

III. BACKGROUND

A. Genetic Programming

GP is an extension of the Genetic Algorithm (GA); which is an evolutionary computation (EC) method proposed by John H. Holland [13]. GP extends the GA to the domain of evolving complete computer programs [14]. Using the Darwinian concepts of natural selection and fitness proportional breeding, populations of programs are genetically bred to solve problems. In tune with the fitness proportional breeding paradigm of GP, a fitness function is required, the fitness function assigns a value to the performance of an individual in the environment (the problem we hope to solve). This value is then used to determine which individuals can breed to produce the members of the next generation. GP has two key advantages over GAs [21]. The first is the flexibility allowed in the representation (of individuals) [21]. The second is interpretability of the evolved programs; these can be easily analysed and decomposed in a meaningful way to understand how GP has learned to solve a problem [20].

B. Tree Adjoining Grammar Guided Genetic Programming

1. Tree Adjoining Grammars

Tree Adjoining Grammars (TAGs) are tree-rewriting systems, originally proposed for natural language processing [14], which give a standard algorithm for converting any Context Free Grammar (CFG) G into a (lexicalised) TAG Glex. Briefly, a TAG system consists of a set of elementary trees $E = I \cup A$ (initial and auxiliary respectively, denoted by α and β). All nodes are labeled grammar symbols, interior labels being restricted to non-terminals. The frontier of an auxiliary tree must contain a 'foot node', with the same label as the root (signified by *). All other non-terminal symbols on the frontier of an elementary tree are available for substitution.

An X-type tree has a root labeled X. TAG systems generate conventional CFG trees (the "derived" tree) from a derivation tree, encoding the history of substitutions and adjunctions generating the derived tree. Starting from a tree at the root, each branch records the address for adjunction or substitution, and the elementary tree used.

Adjunction takes a tree γ with an interior label A, and an A-type auxiliary tree β, producing a new tree γ' by disconnecting the sub-tree α1 rooted at A, attaching β to replace it, and finally re-attaching α1 to the foot node of β.

TAGs have a number of advantages for GP:

- Derivation trees in TAGs are more fine-grained structures than those of CFG.
- They are compact (each elementary node encodes a number of CFG derivations).
- They are closer to a semantic representation.
- Derivation and derived trees provide a natural genotype-to-phenotype map.
- In growing a derivation tree f, one can stop anytime and have a valid derived tree.

We call the last property "feasibility". Feasibility helps TAG3P (described in next subsection) control the exact size of its chromosomes, and also to implement a wide range of genetic operators. A number of them are bio-inspired [10].

2) Tree Adjoining Grammar Guided Genetic Programming

In TAG3P [31], the derivation tree in LTAG (Glex) was used as genotype structure (in which the substitution was used as an in-node operation, and so can be ignored in this paper). The phenotype is the derived tree of Glex, which is a derivation tree in the corresponding context-free grammar G. In [28], it was proven that, when solving typeless problems like GP, there is a one-to-one map between derivation trees in G and expression trees in GP. The mapping schema in our TAG-based representation, therefore, can be summarized in figure 1 as bellow.

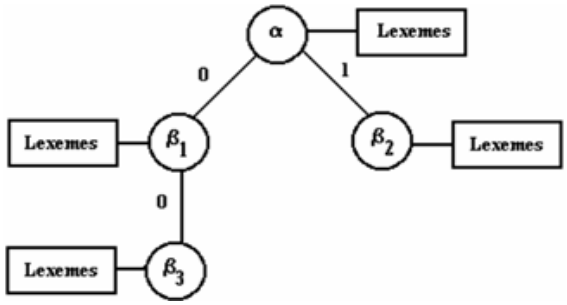


Figure 1. Example of an individual in TAG3P+

Other components of TAG3P are as follows [31]:

Parameters: minimum size of genomes (MIN_SIZE), maximum size of genomes (MAX_SIZE), size of population (POP_SIZE), maximum number of generations (MAX_GEN) and probabilities for genetic operators.

Initialization procedure: Each individual is generated by randomly growing a derivation tree in Glex to a size randomly chosen between MIN_SIZE and MAX_SIZE. Because of TAG feasibility, this always generates valid individuals of exact size.

Fitness Evaluation: an individual derivation is first mapped to the derived CFG tree. The expression defined by the derived tree is then semantically evaluated as in grammar guided genetic programming (GGGP) [28], or translated further into the parse tree and then evaluated as in GP [13].

Main Genetic operators: sub-tree crossover and sub-tree mutation [31].

3) Lexicalized TAGs

Most current linguistic theories give lexical accounts of several phenomena that used to be considered purely syntactic. The information put in the lexicon is thereby increased both in amount and complexity: for example, lexical rules in GPSG [30], HPSG [4], Comhinatory Categoriai Grammars [35], Karttunen's version of Categorial Grammar [16].

We say that a grammar is "lexicalized" if it consists of:

- a finite set of structures associated with each lexical item, which is intended to be the head of these structures;
- an operation or operations for composing the structures. The finite set of structures defines the domain of locality over which constraints are specified, and these are local with respect to their lexical heads.

Context free grammars cannot be lexicalized in general be lexicalized. However TAGs are 'naturally' lexicalized because they use an extended domain of locality [32]. TAGs were first introduced by Joshi [15]. It is known that Tree Adjoining Languages (TALs) are mildly context sensitive. TALs properly contain context-free languages.

A basic component of a TAG is a finite set of elementary trees, each of which defines domain of locality, and can be viewed as a minimal linguistic structure. The elementary structures are projections of lexical items which serve as heads. We recall that tree structures in TAGs correspond to linguistically minimal but complete structures: the complete argument structure in the case of a predicate, the maximal projection of a category in the case of an argument or an adjunct. If a structure has only one terminal, the terminal is the head of the structure; if there are several terminals, the choice of the head for a given structure is linguistically determined, e.g. by the principles of X theory if the structure is of X type. The head of NP is N, that of AP is A. S also has to be considered as the projection of a lexical head, usually V. As is obvious, the head must always be lexically present in all of the structures it produces.

In the TAG lexicon each item is associated with a structure (or a set of structures), and that structure can be regarded as its category, linguistically speaking. Each lexical item has as many entries in the lexicon as it has possible category or argument structures. We will now give some examples of structures that appear in this lexicon.

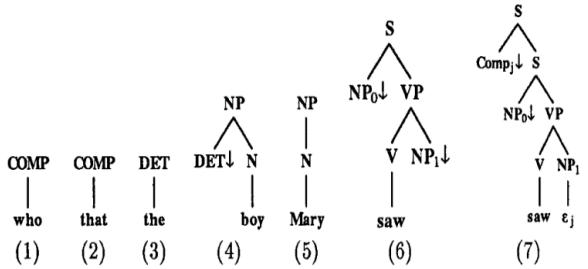


Figure 2. Some examples of initial trees

For $X \neq S$, X-type initial trees correspond to the maximal projection of the category X of the head. They are reduced to a pre-terminal node in the case of simple categories such as COMP or DET (trees 1, 2 and 3) and are expanded into more complex structures in the case of categories taking arguments (tree 4). They correspond to the maximal projection of a category in the case of simple phrases and to trees which will be systematically substituted for one of the argument positions of one of the elementary structures. Trees 6-7 are examples of S-type initial trees: they are usually considered as projections of a verb and usually take nominal complements. The NP-type tree 'Mary' (tree 5), and the NP-type tree 'John' (similar to tree 5), for example, will be inserted by substitution in the tree 6 corresponding to 'NP0 saw NP1' to produce 'John saw Mary'.

4. Surrogate modeling on TAG3P

In this part we will overview about surrogate modeling in TAG3P. The surrogate models will be described in the following:

Step 1. Initialize random populations including rules P

Step 2. Remove duplicate rules and replace them with additional random rules until our population consists of the required number of different rules.

Step 3. Fully evaluate all candidate rules in P are subsequently, this is by far the most time-consuming step in the overall procedure. Once real fitness values are known, we check whether the optimization run should be terminated or continue with another iteration. If we decide to terminate, the best rule found so far is reported as the result of the optimization run.

Step 4. Update our surrogate model to incorporate the new fitness values of all individuals in P.

Step 5. Conventional GP operators [19] for selection, mutation, and crossover, which are used to produce a new, intermediate population P_{imd} taking into account all individuals in P and their fitness values. Tree mutation works by replacing a node of the tree by a new, randomly created subtree. Tree crossover takes two parent trees and produces two offspring trees by exchanging a randomly selected subtree between the two parents.

Step 6. Remove individuals duplicates in Pimd, which are subsequently replaced with additional individuals generated from P in the same way as in step 5. At this stage we have a set of individuals which are produced with a very high probability distinct fitness values. Both P and P_{imd} contain the same number of individuals. As we want to use the surrogate model to select the most promising individuals later on, so we have to create more rules in this step, that is, $|P_{imd}| = n|P|$ with

Step 7. Determine the phenotypic characterization for all rules in P_{imd} following the procedure explained in the next section.

Step 8. Approximate the fitness values for all individuals using the surrogate model. Computing the phenotypic characterization of a rule and evaluating the surrogate for it once is computationally far less

demanding than a full fitness evaluation.
Step 9. Estimated fitnesses are used to select the most promising candidates to form a new population P for the next iteration. Step 3 starts a new iteration of the main optimization loop again, fully evaluating each individual in P using the expensive fitness function.

IV. EXPERIMENTS AND RESULTS

This section presents the experimental setting in this paper and the results of applying Surrogate model on TAG3P to generate new rules for IDS.

C. Experimental settings

In this paper, we propose an approach based on TAG3P with Surrogate model to detect known and unknown attacks on the network. In this method, we use the parse tree to represent the initial population as the rule. These rules are selected on the basis of knowledge of the attacks in DARPA dataset namely: name lyland, synflood, ping of death (pod), Smurf, Teardrop, Back, Neptunes, ip_sweep, sweep port, and UDP storm attacks.
Individual solution in a population is represented as a derivation tree that we describe using a string data structure. For example, a tree can be represented as "AabAcdAceI". A means "and" operator; a, b, c, d, and e correspond to the conditions in the rules; I is the consequence, which means intrusion. The redundant conditions in the rule will be deleted after the evolution, and thus, AabAcdAceI can be interpreted as "if a and b and c and d and e then intrusion". The attribute values of a, b, c, d, e are selected from known attacks. The value of attributes will be represented in tabe 1.

Table 1. Representation of attributes value

Terminal	Meaning	Terminal	Meaning
s	num compromised = 0	i	protocol type = udp
b	count > R1	f	land = 1
c	srv count > R1	o	land = 0
d	dst host count > R1	j	count < R2
k	srv count < R2	q	synflood = 0
p	wrong fragment = 0	r	synflood > 1
e	dst host srv count > R1	a	protocol type = tcp
l	dst host count < R2	h	protocol type = icmp
m	dst host srv count < R2	u	same srv rate = 0.00
t	num compromised > 1	v	same srv rate = 1.00
g	wrong fragment > 1	w	diff srv rate > R3
Note: R1, R2, and R3 are random.			

New rules are generated in two phases. In the first step, temporary new rules are composed of new rules generated by four operators including mutation, reproduction, crossover, and dropping condition and additional rules directly generated from previous populations. Thus, the number of temporary new rules is doubled. In the second phase, some of the temporary new rules with the highest fitness scores after token competition are retained and passed to the next generation.
To assess the feasibility and efficiency of GP for intrusion detection, we have selected an initial population of 30 rules that cover a series of network-based attacks. Table 2 shows some instances of the initial rules in our experiments.

Table 2. Some of Init Rules

Rules	Meaning
Afp	if land=1 and wrong_fragment=0 then intrusion
AAgg	if wrong_fragment > 1 then intrusion
Aab	if protocol_type=tcp and count>3 then intrusion
bAbA	if srv_count>3 then intrusion
Aigq	if protocol_type=udp and wrong_fragment>1 and synflood>1 then intrusion
Aaav	if protocol_type=tcp and same_srv_rate=1.00 then intrusion
AqsujaA	if protocol_type=tcp and count<3 and synflood=0 and num_comperomised=0 and same_srv_rate=0.00 then intrusion
Aajt	if count<3 and num_compromised>1 then intrusion
Avfrg	if land = 1 and wrong fragment > 1 and same srv rate = 1.00 and synflood > 1 then intrusion

Each generation, we calculate the fitness value for each rule based on the training dataset. After that, we pick out five individuals (rules) best fitness value into the next generation and then perform genetic manipulations for the remaining individuals in the population next. We noticed that five individuals selected in the previous generation will certainly continue to exist in the next generation, this would significantly reduce unnecessary calculations and increase the speed of algorithm.
The rules in the initial population are evolved using mutation, crossover, and dropping condition operators. The rates of crossover, mutation, and dropping condition operations are respectively 0.9, 0.01, and 0.001 for each rule. Table 3 shows the parameter settings on our experiments. Forty offspring rules are evolved from the previous forty parent rules. Based on token competition, combining offspring rules with parent rules generates temporary new rules. Some of the temporary new rules with highest fitness scores after token competition are selected as the new rules.

D. Rules Evaluation

Fitness functions ensure that the evolution is toward optimization by calculating the fitness value for each individual (rule) in the population. The fitness value evaluates the performance of each individual in the population. An evaluation (fitness) function based on the support-confidence framework is developed as the fitness function in our rule learning approach. Support measures the coverage of a rule, it is a ratio of the number of records covered by the rule to the total number of records. Confidence factor (cf) is the confidence of the consequent to be true under the antecedents, and is just the same as the rule accuracy. It is the ratio of the number of records matching both

the consequent and the antecedents to the number of records matching only the antecedents. For a rule "if A then B" and with a training set of N cases, we define:

$$support = \frac{|A \quad and \quad B|}{|N|}; \quad and \quad cf = \frac{|A \quad and \quad B|}{|A|}$$

In the evaluation process, each rule is checked with every record in the training set. Three statistics are counted. antes_hit is the number of records matching the antecedents (the 'if part), consq_hit is the number of records that match the consequent (the 'then' part), and both_hit is the number of records that match the whole rule (both the 'if and the 'then' parts).

The cf is the fraction both_hit/antes_hit. But a rule with a high confidence factor does not mean that it behaves significantly different from the average. Therefore we need to consider the average probability of consequent (prob). The value prob is equal to consq_hit/total, where total is the total number of records in the training set. This value measures the confidence for the consequent under no particular antecedent.

A formula similar to the likelihood ratio is used to define the normalized confidence factor normalized_cf:

$$normalized_\text{cf} = cf \times \log\left(\frac{cf}{prob}\right)$$

The log function measures the order of magnitude of the ratio cf/prob. The normalized value is a product of two factors: cf and log(cf/prob). A high value of normalized_cf requires simultaneously a high value on the rule confidence factor (cf) and a high value on the rule cf over the average probability (cf/prob). We define, $cf = \frac{|A \quad and \quad B|}{|A|}$, and $prof = \frac{|B|}{N}$; If $\frac{|A \quad and \quad B|}{|A|} = \frac{|B|}{N}$ then $\frac{cf}{prof} = 1$ and normalized_cf=0. The value cf monotonically increases with |A and B| and monotonically decreases with |A|. The value prob monotonically increases with |B| and thus normalized_cf monotonically decreases with |B|.

Support is another measure that we need to consider. A rule can have a high accuracy but the rule may be formed by chance and based on a few training examples. This kind of rules does not have enough support.

The value support is defined as both_hit/total. If support is below a user-defined minimum threshold (min_support), the confidence factor of the rule should not be considered. This can avoid the waste of effort to evolve those rules with a high confidence but cannot be generalized.

Finally, We define our fitness function to be:

$$raw_fitness = \begin{cases} cf + support & if \quad \text{support} > \text{min_support} \\ support & otherwise \end{cases}$$

where, we set the min_support in our experiment as 0.001

In intrusion detection, four metrics are typically used in order to quantify the performance of the IDS for each rule:

+) Detection rate (DR):

$$DR = \frac{TPRate}{(TPRate + FNRate)}$$

+) True positive rate (TPRate):

$$TPRate = \frac{NumberofAttackDetection}{TotalAttack}$$

+) False positive rate (FPRate):

$$FPRate = \frac{FP}{FP + TN}$$

+) False negative rate (FNRate): fraction of intrusions incorrectly diagnosed (not detected)

$$FNRate = 1 - TPRate$$

where, FP is number of false possitives and TN is number of true negatives.

E. Results and Discussion

Using the data generated as described above, we conducted 1000 runs of the IDS, each run differing only in the random seeds that are used for the initial population creation. Each run will create new rule that corresponds to the best fitness value. The control parameters used during the running of the TAG3P are shown in Table 3 and were chosen due to work in [19].

Table 3. Parameter settings for TAG3P

Parameter	Settings
Population size	200
Function set	AND
Terminal set	a, b, c, d…x, y, z… the parameter and condition in each rule
Crossover probability	0.9
Mutation Probability	0.01
Tournament size	4
Generations	51

The evolution will not be terminated until we have executed 1,000 runs or the fitness value for each rule is bigger than or equal a threshold to get the best rule, the threshold is adjust from 0.90 to 0.97.

Detection of attacks involved in the test dataset, and not occurring in the training dataset, assesses the potential ability to detect novel attacks. To evaluation new rules, we use KDD'99 Datasets. We use four performance metrics to evaluate the new rules, namely DR, detection rate (DR), true positive rate (TPRate), false positive rate (FPRate), and false negative rate (FNR). A false positive occurs when a rule classifies normal traffic as intrusive. A false negative occurs when a rule characterizes an intrusion as normal. For each rule, we calculate its DR, TPRate, FPRate, and FNRate independently. We find that every time we use SuGP to evolve the rules, the number of generated rules is different and thus its DR, TPRate, FPRate, and

FNRate for each rule are also different. Therefore, to statistically evaluate the efficiency of our TAG3P-based approach, its DR, TPRate, FPRate, and FNRate are defined as the arithmetical average of the sum of all new rules' rates.

The rules in the initial population are evolved using mutation, crossover, and dropping condition operators. The rates of crossover and mutation operations are respectively 0.9, and 0.01 for each rule. Offspring rules are evolved from the previous parent rules. Based on token competition, combining offspring rules with parent rules generates new rules. We will choose 5 rules from population of new rules with highest fitness scores after token competition are selected as the new rules.

New rules are generated in two phases. In the first step, new rules are composed of new rules generated by four operators including mutation, reproduction, crossover, and dropping condition and additional rules directly generated from previous populations. In the second phase, we choose five new rules with the highest fitness scores after token competition are retained and passed to the next generation.

Since new rules are generated, we use them to detect intrusion on the KDD'99 datasets to show detection rate of each rule on datasets. We execute 100,000 run and with each run we will evaluate its DR, TPR, FPR, and FNR of each rule, after that we will calculate the average of all rules. We show that the average value of four performance metrics with two cases:

Case 1: When we adjust the threshold value for fitness value increase to get the detection rate is 100%, the true positive rate is only 90.43%, the false positive rate is 9.57% and the false negative rate is zero.

Case 2: When we adjust the threshold value for fitness value decrease until the detection rate is 97.5%, we receive the true positive rate was only 99.2%, the false positive rate is 0.8% and the false negative rate is 0.5%.

Table 4. Result of Detection

Performance metrics	Case 1	Case 2
DR	100%	97.5%
TPRate	90.43%	99.2%
FPRate	9.57%	0.8%
FNRate	0%	0.5%

The result in table 4 shows that, when we adjust the threshold for fitness value of each rule will get the change of detection rates. The detection rate will high when threshold values is high, but true positive rate is decreased and the false positive rate is increases. Otherwise, when the threshold value adjustment decrease will reduce the detection rate, however, will increase the true positive rate and decrease the false positive rate. By reducing the detection rate has led pass some intrusion in dataset.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented and evaluated a Surrogate model on TAG3P-based approach for detecting known or novel attacks on a network. The proof of concept implementation shows that new rules generated by our approach have the potential capability to detect attacks. Experiments showed that when adjusting the threshold value for the fitness value of each new rule created will change the performance of intrusion detection with this rule. Therefore, we should adjust threshold value to make effect better. Besides choosing the number of new laws have the highest fitness scores will affect the speed of creating new laws as well as the effectiveness of the law in intrusion detection. In our experiments, we selected 5 of law has the highest fitness scores to take into the next generation.

In the near future, we aim to apply self-adjust threshold value for the fitness value of each rule to find the best rule of each generation. We argue that when applying the self-adjust threshold value will enhance its performance significantly. In future research, we will conduct and examine this hypothesis.

REFERENCES

[1] A J.P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical Report, James P. Anderson Co., Fort Washington, PA, 1980.

[2] Branke, J., Hildebrandt, T., and Scholz-Reiter, B. (2014). Hyper-heuristic evolution of dispatching rules—A comparison of rule representations. Evolutionary Computation

[3] Emmerich, M., Giotis, A., Ozdemir, M., Back, T., and Giannakoglou, K. (2002). Metamodel-assisted evolution strategies. Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, pp. 361–370.

[4] Gazdar, G.; Klein, E.; Pullum, G. K.; and Sag, I. A., 1985. Generalized Phrase Structure Grammars. Blackwell Publishing, Oxford. Also published by Harvard University Press, Cambridge, MA.

[5] Jonatan Gomez and Dipankar Dasgupta. Evolving fuzzy classifiers for intrusion detection. In Proceedings of the 2002 IEEE Workshop on Information Assurance, West Point, NY, USA, 2002.

[6] J. Gomez and D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection", Proceedings of the IEEE, 2005.

[7] R. H. Gong, M. Zulkernine, P. Abolmaesumi, "A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection", 2005.

[8] Gustafson, S. (2004). An analysis of diversity in genetic programming. Ph.D. thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, England.

[9] M. M. M. Hassan, "Current Studies on Intrusion Detection System, Genetic Algorithm and Fuzzy Logic", International Journal of Distributed and Parallel Systems, Vol. 4, No. 2, pp. 35-47, 2013.

[10] Nguyen Xuan Hoai, McKay R.I., and Abbass H.A. (2003): Tree Adjoining Grammars, Language Bias, and Genetic Programming. In Proceedings of EuroGP 2003, Ryan C. et al (Eds), LNCS 2610, Springer Verlag, pp. 335-344.

[11] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, 6(3): 151-180, August 1998.

[12] Jin, Y., Olhofer, M., and Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation, 6(5): 481–494.

[13] Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation, 1(2): 61–70.

[14] Joshi A. K., Levy L. S., and Takahashi M.: Tree Adjunct Grammars. Journal of Computer and System Sciences, 10 (1), (1975) 136-163.
[15] Joshi, Aravind K., 1985. How Much Context-Sensitivity is Necessary for Characterizing Structural Descriptions Tree Adjoining Grammars. In DoTty, D.;
[16] Karttunen, Lauri, 1986. Radical Lexicalism. Technical Report CSLI-86-68, CSLI, Stanford University. To also appear in New Approaches to Phrase Structures, University of Chicago Press, Baltin, M. and Kroch A., Chicago, 1988.
[17] Jungwon Kim, Peter J. Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross. Immune system approaches to intrusion detection - a review. Natural Computing, 6(4): 413-466, December 2007.
[18] Kim, Y.-H., Moraglio, A., Kattan, A., and Yoon, Y. (2014). Geometric generalisation of surrogate model-based optimisation to combinatorial and program spaces. Mathematical Problems in Engineering.
[19] Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection. Cambridge, MA: MIT Press.
[20] Koza, J. R., and Re, D., Bennett III, F. H., and Keane, M (1999). Genetic Programming 3: Darwinian Invention and Problem Solving. Morgan Kaufman.
[21] Kozza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., YU, J., and Lanza, G (2003). Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers.
[22] Pavel Laskov, Patrick Dssel, Christin Schfer, and Konrad Rieck. Learning intrusion detection: Supervised or unsupervised? In Image Analysis and Processing ICIAP 2005, volume 3617 of Lecture Notes in Computer Science, pages 50-57. Springer Berlin Heidelberg, 2005.
[23] Wenke Lee and Salvatore J. Stolfo. Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium - Volume 7, SSYM'98, pages 6-6, Berkeley, CA, USA, 1998.
[24] Wei Li. Using genetic algorithm for network intrusion detection. In Proceedings of the US DoE Cybersecurity Conference, Kansas City, KS, USA, 2004.
[25] Lim, D., Jin, Y., Ong, Y.-S., and Sendhoff, B. (2010). Generalizing surrogate-assisted evolutionary computation. IEEE Transactions on Evolutionary Computation, 14(3):329–355.
[26] Lippmann, R., et al., The 1999 DARPA Off-Line lnhusion Detection Evaluation. Computer Networks, 34(4) 579-595, 2000.
[27] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In Proceedings of the 2002 International Joint Conference on Neural Network (IJCNN), volume 2, pages 1702-1707, 2002.
[28] Nguyen, S., Zhang, M., Johnston, M., and Tan, K.-C. (2014). Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Transactions on Evolutionary Computation, 18(2):193–208
[29] Truong Son Pham, Tuan Hao Hoang and Van Canh Vu, Machine learning techniques for web intrusion detection - a comparison, 2016 Eighth International Conference On Knowledge And Systems Engineering, Hanoi, Vietnam, 06 - 08 October 2016
[30] Pollard, Carl and Sag, Ivan A., 1987. Information-Based Syntax and Semantics. Vol 1: Fundamentals. csli.
[31] Rattle, A. (1998). Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. Parallel problem solving from nature (pp. 87–96). Berlin: Springer-Verlag.
[32] Schabes, Yves and Joshi, Aravind K., June 1988. An Earley-Type Parsing Algorithm for Tree Adjoining Grammars. In 26 th Meeting of the Association for Computational Linguistics. Buffalo.
[33] Chris Sinclair, Lyn Pierce, and Sara Matzner. An application of machine learning to network intrusion detection. In Proceedings of the 15th Annual Computer Security Applications Conference, ACSAC '99, Washington, DC, USA, 1999.
[34] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the IEEE Symposium on Security and Privacy, 2010.
[35] Steedman, M., 1988. Combinatory Grammars and Parasitic Gaps. To appear in Natural Language and Linguistic Theory.
[36] T. Xia, G. Qu, S. Hariri, M. Yousif, “An Efficient Network Intrusion Detection Method Based on Information Theory and Genetic Algorithm”, Proceedings of the 24th IEEE International Performance Computing and Communications Conference (IPCCC ’05), Phoenix, AZ, USA, 2005.
[37] Yao, J. T., S.L. Zhao, and L.V. Saxton, “A Study On Fuzzy Intrusion Detection”, Proceedings of the Data Mining, Intrusion Detection, Information Assurance, And Data Networks Security”, SPIE, Vol. 5812, Orlando, Florida, USA, pp. 23-30, 2005.
[38] Mahdi Zamani, Mahnush Movahedi, Mohammad Ebadzadeh, and Hossein Pedram. A DDoS-aware IDS model based on danger theory and mobile agents. In Proceedings of the 2009 International Conference on Computational Intelligence and Security - Volume 01, CIS '09, pages 516-520, Washington, DC, USA, 2009. IEEE Computer Society.
[39] Mahdi Zamani, Mahnush Movahedi, Mohammad Ebadzadeh, and Hossein Pedram. A danger-based approach to intrusion detection. CoRR, abs/1401.0102, 2014.

★★★