Evolutionary Design of Intrusion Detection Programs

Ajith Abraham[1,3], Crina Grosan[2], and Carlos Martin-Vide[3]
(Corresponding author: Ajith Abraham)

IITA Professorship Program, School of Computer Science and Engineering, Yonsei University[1]
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Republic of Korea
(e-mail: ajith.abraham@ieee.org)
Department of Computer Science, Babes-Bolyai University[2]
Kogalniceanu 1, Cluj-Napoca 400084, Romania
Research Group on Mathematical Linguistics (GRLMC), Rovira i Virgili University, Spain[3]

(Received Oct. 16, 2005; revised and accepted Nov. 25, 2005 & Jan. 27, 2006)

Abstract

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. This paper proposes the development of an Intrusion Detection Program (IDP) which could detect known attack patterns. An IDP does not eliminate the use of any preventive mechanism but it works as the last defensive mechanism in securing the system. Three variants of genetic programming techniques namely Linear Genetic Programming (LGP), Multi-Expression Programming (MEP) and Gene Expression Programming (GEP) were evaluated to design IDP. Several indices are used for comparisons and a detailed analysis of MEP technique is provided. Empirical results reveal that genetic programming technique could play a major role in develop- ing IDP, which are light weight and accurate when compared to some of the conventional intrusion detection systems based on machine learning paradigms.

Keywords: Intrusion detection program, genetic programming, machine learning, light weight intrusion detection system

1 Introduction

Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability [37]. Confidentiality (or secrecy) means that information is disclosed only according to policy, integrity means that information is not destroyed or corrupted and that the system performs correctly, availability means that system services are available when they are needed. Security threats come from different sources such as natural forces (such as flood), accidents (such as fire), failure of services (such as power) and people known as intruders. There are two types of intruders: the external intruders who are unauthorized users of the machines they attack, and internal intruders, who have permission to access the system with some restrictions. The traditional prevention techniques such as user authentication, data encryption, avoiding programming errors and firewalls are used as the first line of defense for computer security. If a password is weak and is compromised, user authentication cannot prevent unauthorized use, firewalls are vulnerable to errors in configuration and ambiguous or undefined security policies. They are generally unable to protect against malicious mobile code, insider attacks and unsecured modems. Programming errors cannot be avoided as the complexity of the system and application software is changing rapidly leaving behind some exploitable weaknesses. Intrusion detection is therefore required as an additional wall for protecting systems. Intrusion detection is useful not only in detecting successful intrusions, but also provides important information for timely countermeasures.

An intrusion is defined [20] as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. This includes a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable. An attacker can gain illegal access to a system by fooling an authorized user into providing information that can be used to break into a system. An attacker can deliver a piece of software to a user of a system which is actually a trojan horse containing malicious code that gives the attacker system access. Bugs in trusted programs can be exploited by an attacker to gain unauthorized access to a computer system. There are legitimate actions that one can perform that when taken to the extreme can lead to system failure. An attacker can gain access because of an error in the configuration of a system. In some cases it is possible to fool a system into giving access by misrepresenting oneself. An example is sending a TCP packet that has a forged source address that makes the packet appear to come from a trusted host. Intrusions are classified [38] as six types. Attempted break-ins, which are detected by typical behavior profiles or violations of security constraints. Masquerade attacks, which are detected by atypical behavior profiles or violations of security constraints. Penetration of the security control system, which are detected by monitoring for specific patterns of activity. Leakage, which is detected by atypical use of system resources. Denial of service, which is detected by a typical use of system resources. Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

This paper attempts to illustrate how genetic programming techniques could be deployed for detecting known types of attacks. In this respect, three genetic programming techniques are used in the experiments. We considered attack types of varying difficulties. Results are analyzed by considering several measures like classification accuracy, false alarm rate and graphical comparisons with the target results. As evident from the results, genetic programming techniques are very appealing for developing light weight (the developed IDS is only a software fragment) intrusion detection systems.

The paper is organized as follows: Section 2 presents the related research done in the area of intrusion detection. In the Section 3, the three genetic programming techniques applied (LGP, MEP and GEP) are described. Section 4 contains the set of experiments, discussions and analysis of results. Section 5 presents conclusions of this research.

2 Related Research

James Anderson [3] first proposed that audit trails should be used to monitor threats. All the available system security procedures were focused on denying access to sensitive data from an unauthorized source. Dorothy Denning [13] later proposed the concept of intrusion detection as a solution to the problem of providing a sense of security in computer systems. The basic idea is that intrusion behavior involves abnormal usage of the system. The model is a rule-based pattern matching system. Some models of normal usage of the system could be constructed and verified against usage of the system and any significant deviation from the normal usage flagged as abnormal usage.

Statistical approaches compare the recent behavior of a user of a computer system with observed behavior and any significant deviation is considered as intrusion. This approach requires construction of a model for normal user behavior. Predictive pattern generation uses a rule base of user profiles defined as statistically weighted event sequences [39]. This method of intrusion detection attempts to predict future events based on events that have already occurred.

State transition analysis approach construct the graphical representation of intrusion behavior as a series of state changes that lead from an initial secure state to a target compromised state. Using the audit trail as input, an analysis tool can be developed to compare the state changes produced by the user to state transition diagrams of known penetrations [21]. Keystroke monitoring technique utilizes a user's keystrokes to determine the intrusion attempt. The main approach is to pattern match the sequence of keystrokes to some predefined sequences to detect the intrusion.

Model-Based approach attempts to model intrusions at a higher level of abstraction than audit trail records. This allows administrators to generate their representation of the penetration abstractly, which shifts the burden of determining what audit records are part of a suspect sequence to the expert system. This technique differs from the rule-based expert system technique, which simply attempt to pattern match audit records to expert rules [18].

The pattern matching [24] approach encodes known intrusion signatures as patterns that are then matched against the audit data. Intrusion signatures are classified using structural interrelationships among the elements of the signatures. The patterned signatures are matched against the audit trails and any matched pattern can be detected as an intrusion.

During recent years, several data mining approaches have been also used to construct IDS [25, 30]. Depren et al. [14] proposed a novel IDS architecture utilizing both anomaly and misuse detection. The hybrid IDS architecture consists of an anomaly detection module, a misuse detection module and a decision support system combining the results of these two detection modules. The proposed anomaly detection module uses a Self-Organizing Map (SOM) structure to model normal behavior. The proposed misuse detection module uses decision tree algorithm to classify various types of attacks. A rule-based Decision Support System (DSS) is also developed for interpreting the results of both anomaly and misuse detection modules.

Chen et al. [10] suggested two data mining methodologies involving artificial neural networks (ANN) and support vector machine (SVM) ([40]) and two encoding methods namely simple frequency-based scheme and tfidf scheme to detect potential system intrusions. Their experiments show that SVM with tfidf scheme achieved the best performance, while ANN with simple frequency based scheme achieved the worst.

Dasgupta et al. [12], presented a security agent architecture, which is useful as an administrative tool for intrusion detection. The agent-based monitoring and detection system, could detect malfunctions, faults, abnormalities, misuse, deviations, intrusions, and provide recommendations (in the form of common intrusion detection language).

Zhang and Shen [42] have formulated intrusion detection as a binary classification problem, using SVM and additionally, some text processing techniques are also employed for intrusion detection, based on the characterization of the frequencies of the system calls executed by the privileged programs.

Rohrmair and Lowe [34] demonstrate the modelling and analysis of IDS using the process algebra communicating sequential processes and its model checker FDR. Authors show that this analysis can be used to discover attack strategies that can be used to blind an IDS, even a hypothetically perfect one that knows all the weaknesses of its protected host.

Network-based intrusion detection systems (NIDSs) frequently have problems with handling heavy traffic loads in real-time, which result in packet loss and false negatives. Jiang et al. [22] present a high-performance network IDS, called HPMonitor, which combines a high-efficiency detection engine and a load-balancing device to address these problems. HPMonitor uses a flow-based dynamic load-balancing algorithm called dynamic least load first (DLLF) algorithm, and introduces a new multi-pattern string matching algorithm called shift max algorithm (SMA). The test results reveal that the DLLF algorithm is an effective balancing algorithm for NIDS.

Current IDS examine all data features to detect intrusion or misuse patterns. Some of the features may be redundant or contribute little (if anything) to the detection process. Chebrolu et al. [9] investigated the performance of two feature selection algorithms involving Bayesian networks (BN) and Classification and Regression Trees (CART) [7] and an ensemble of BN and CART. Empirical results indicate that significant input feature selection is important to design an IDS that is lightweight, efficient and effective for real world detection systems.

Most IDS have a single-level structure can only detect either misuse or anomaly attacks. Some IDSs with multi-level structure or multi-classifier are proposed to detect both attacks, but they are limited in adaptive learning. Zhang et al. [41] proposed a serial hierarchical IDS (SHIDS)to identify misuse attacks accurately and anomaly attacks adaptively.

Beghdad [4] introduces a novel anomaly intrusion detection method based on a Within-Class Dissimilarity (WCD). This approach functions by using an appropriate metric WCD to measure the distance between an unknown user and a known user defined respectively by their profile vectors.

Boukerche et al. [8] propose an IDS technique based upon data analysis inspired by the natural immune human systems. Authors illustrated how the proposed scheme extracts salient features of the immune human system and maps them within a software package designed to identify security violations of a computer system and unusual activities according to the usage log files.

Scot [36] proposes latent variable hierarchical model construction using Bayesian methods which leads to coherent systems that can handle the complex distributions involved with network traffic. Bayes' rule provides a means of combining competing intrusion detection methods such as anomaly detection and pattern recognition. Bayesian methods present evidence of intrusion as probabilities, which are easy for human fraud investigators to interpret. Hierarchical approach allows transactions to communicate information about possible intrusions across time and accounts.

Support vector machines (SVM) have proven to be a good candidate for intrusion detection because of its training speed and scalability [27].

Fuzzy logic has proved to be a powerful tool for decision making to handle and manipulate imprecise and noisy data. Three different types of fuzzy classifiers have been used for intrusion detection. The first classifier uses a histogram to generate an antecedent membership function and each attribute is partitioned into several fuzzy sets. Second method uses a rule generation based on partition of overlapping areas. The third method uses a neuro-fuzzy computing framework in which a fuzzy inference system is learned using neural network learning paradigms [30].

Multivariate Adaptive Regression Splines (MARS) is an innovative approach that automates the building of accurate predictive models for continuous and binary dependent variables [17]. It excels at finding optimal variable transformations and interactions, and the complex data structure that often hides in high-dimensional data.

The swarm intelligence algorithm fully uses agents that stochastically move around the classification habitat following pheromone concentrations. Having that aim in mind, a self-organized ANT colony based Intrusion Detection System (ANTIDS) is used to cluster the intrusion patterns [33].

Decision tree induction is one of the classification algorithms in the data mining. Classification algorithm is inductively learned to construct a model from the pre-classified data set. The inductively learned model of classification algorithm is used to develop IDS [7]. Several hybrid approaches for modelling IDS have been also explored. Decision Trees (DT) and Support Vector Machines (SVM) are combined as a hierarchical hybrid intelligent system model (DT-SVM) [30].

An IDS based on general and enhanced Flexible Neural Tree (FNT) is explored by Chen et al. [11]. Based on the pre-defined instruction/operator sets, a flexible neural tree model can be created and evolved. The FNT structure is developed using an evolutionary algorithm and the parameters are optimized by particle swarm optimization algorithm.

3 Genetic Programming Techniques for Intrusion Detection

Genetic Programming (GP) technique provides a framework for automatically creating a working computer program from a high-level problem statement of the problem. Genetic programming achieves this goal of automatic programming by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations. GP is the extension of evolutionary learning into the space of computer programs.

A population of candidate solutions (intrusion detection programs) is initialized. New solutions are created by applying reproduction operators (mutation and/or crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation.

3.1 Linear Genetic Programming (LGP)

Linear Genetic Programming [5, 6] is a variant of the GP technique that acts on linear genomes. Its main characteristics in comparison to tree-based GP lies in that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like c/c ++). The basic unit of evolution here is a native machine code instruction that runs on the floating-point processor unit (FPU). LGP uses a specific linear representation of computer programs. Instead of the tree-based GP expressions of a functional programming language (like *LISP*) programs of an imperative language (like *C*) are evolved. A LGP individual is represented by a variable-length sequence of simple *C*language instructions. Instructions operate on one or two indexed variables (registers) r, or on constants c from predefined sets.

The result is assigned to a destination register, for example, $ri = rj* c$.

A sample LGP program is illustrated below:

```
void LGP(double v[8])
 v[0] = v[5] + 73;
 v[7] = v[3] – 59;
 if (v[1] >0)
 if (v[5] >21)
        v[4] = v[2] .v[1];
        v[2] = v[5] + v[4];
        v[6] = v[7] .25;
        v[6] = v[4] – 4;
        v[1] = sin(v[6]);
 if (v[0] > v[1])
        v[3] = v[5] .v[5];
        v[7] = v[6] .2;
        v[5] = v[7] + 115;
 if (v[1] <= v[6])
        v[1] = sin(v[7]);
```

An LGP chromosome can be turned into a functional representation by successive replacements of variables starting with the last effective instruction. The maximum number of symbols in a LGP chromosome is 4 * Number of instructions.

An important LGP parameter is the number of registers used by a chromosome. The number of registers is usually equal to the number of attributes of the problem. LGP uses a modified steady-state algorithm. The initial population is randomly generated. The following steps are repeated until a termination criterion is reached: Four individuals are randomly selected from the current population. The best two of them are considered the winners of the tournament and will act as parents. The parents are recombined and the offspring are mutated and then replace the losers of the tournament.

The settings of various linear genetic programming system parameters are of utmost importance for successful performance of the system. The population space has been subdivided into multiple subpopulation or demes. Migration of individuals among the subpopulations causes evolution of the entire population. It helps to maintain diversity in the population, as migration is restricted among the demes. Moreover, the tendency towards a bad local minimum in one deme can be countered by other demes with better search directions.

The various LGP search parameters are the mutation frequency, crossover frequency and the reproduction frequency. The crossover operator acts by exchanging sequences of instructions between two tournament winners. Steady state genetic programming approach was used to manage the memory more effectively.

3.2 Multi Expression Programming (MEP)

A GP chromosome generally encodes a single expression (computer program). By contrast, a Multi Expression Programming (MEP) chromosome encodes several expressions [28, 29]. The best of the encoded solution is chosen to represent the chromosome by supplying the fitness of the individual.

MEP genes are represented by substrings of a variable length. The number of genes per chromosome is constant. This number defines the length of the chromosome. Each gene encodes a terminal or a function symbol. A gene that encodes a function includes pointers towards the function arguments. Function arguments always have indices of lower values than the position of the function itself in the chromosome.

An example of a chromosome using the sets $F= \{+, *\}$ and $T= \{a, b, c, d\}$ is given below:

1: a
2: b
3: + 1, 2
4: c
5: d
6: + 4, 5
7: * 3, 6

The maximum number of symbols in the MEP chromosome is given by:

$$number\ of\ symbols = (n+1)(number of genes - -1) + 1,$$

where n is the number of arguments of the function with the greatest number of arguments. The maximum number of effective symbols is achieved when each gene (except the first one) encodes a function symbol with the highest number of arguments. The minimum number of effective symbols is equal to the number of genes and it is achieved when all genes encode terminal symbols only.

The translation of a MEP chromosome into a computer program represents the phenotypic transcription of the MEP chromosomes. Phenotypic translation is obtained by parsing the chromosome top-down. A terminal symbol specifies a simple expression. A function symbol specifies a complex expression obtained by connecting the operands specified by the argument positions with the current function symbol.

For instance, Genes 1, 2, 4 and 5 in the previous example encode simple expressions formed by a single terminal symbol. These expressions are:

$$E_1 = a,$$
$$E_2 = b,$$
$$E_4 = c,$$
$$E_5 = d.$$

Gene 3 indicates the operation + on the operands located at Positions 1 and 2 of the chromosome. Therefore Gene 3 encodes the expression: $E_3 = a + b$. Gene 6 indicates the operation + on the operands located at Positions 4 and 5. Therefore Gene 6 encodes the expression: $E_6 = c + d$. Gene 7 indicates the operation * on the operands located at Positions 3 and 6. Therefore Gene 7 encodes the expression: $E_7 = (a + b) * (c + d)$. E_7 is the expression encoded by the whole chromosome. Due to its multi expression representation, each MEP chromosome may be viewed as a forest of trees rather than as a single tree, which is the case of GP.

3.3 Gene Expression Programming

The individuals of GEP [15, 16] are encoded as linear chromosomes which are expressed or translated into expression trees (branched entities). Thus, in GEP, the genotype (the linear chromosomes) and the phenotype (the expression trees) are different entities (both structurally and functionally) that, nevertheless, work together forming an indivisible whole.

In contrast to its analogous cellular gene expression, GEP is rather simple. The main players in GEP are only two: the chromosomes and the Expression Trees (ETs), being the latter the expression of the genetic information encoded in the chromosomes. GEP uses linear chromosomes that store expressions in breadth-first form. A GEP gene is a string of terminal and function symbols. GEP genes are composed of a *head* and a *tail*. The head contains both function and terminal symbols. The tail may contain terminal symbols only. For each problem the head length (denoted h) is chosen by the user. The tail length (denoted by t) is evaluated by:

$$t = (n-1)h + 1,$$

where n is the number of arguments of the function with more arguments. Let us consider a gene made up of symbols in Set S:

$$S = \{*, +, -, a, b\}.$$

In this case $n = 2$. If we choose $h = 10$, then we get $t = 11$, and the length of the gene is $10 + 11 = 21$. Such a gene is given below:

$$C_{GEP} = + * ab - +aab + ababbbababb.$$

The *expression* encoded by the gene C_{GEP} is:

$$E = a + b * ((a + b) - a).$$

GEP genes may be linked by a function symbol in order to obtain a fully functional chromosome. In the current version of GEP the linking functions for algebraic expressions are addition and multiplication. A single type of function is used for linking multiple genes.

The initial population is randomly generated. The following steps are repeated until a termination criterion is reached: A fixed number of the best individuals enter the next generation (elitism). The mating pool is filled by using binary tournament selection. The individuals from the mating pool are randomly paired and recombined. Two offspring are obtained by recombining two parents. The offspring are mutated and they enter the next generation.

4 Experiment Setup and Results

We performed five experiments using five different test data. The data for our experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Labs [26]. The data set has 41 attributes for each connection record plus one class label as given in Table 1. The data set contains 24 attack types that could be classified into four main categories:

DoS: Denial of Service

Denial of Service (DoS) is a class of attack where an attacker makes a computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine.

R2L: Unauthorized Access from a Remote Machine

A remote to user (R2L) attack is a class of attack where an attacker sends packets to a machine over a network, then exploits the machine's vulnerability to illegally gain local access as a user.

U2R: Unauthorized Access to Local Super User (root)

User to root (U2Su) exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system.

Probing: Surveillance and Other Probing

Probing is a class of attack where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits.

Experiments presented in this paper have two phases namely training and testing. In the training phase, LGP, MEP and GEP models were constructed using the training data to give maximum generalization accuracy on the unseen data. The test data is then passed through the saved trained model to detect intrusions in the testing phase. The 41 features are labelled as shown in Table 1 and the class label is named as *AP*.

This data set has five different classes namely *Normal, DoS, R2L, U2R* and *Probes.* The training and test comprises of 5,092 and 6,890 records respectively [23]. All the training data were scaled to (0-1). Using the data set, we performed a 5-class classification. The normal data belongs to Class 1, probe belongs to Class 2, denial of service belongs to Class 3F, user to super user belongs to Class 4, remote to local belongs to Class 5.

4.1 Parameters Settings

The various parameter settings for LGP is depicted in Table 2 [1]. Parameters used by MEP are presented in Table 3 [19]. We made use of *+, - , *, /, sin, cos, sqrt, ln, lg,* log_2*, min, max,* and *abs* as function's set. Parameters used by GEP are depicted in Table 4.

4.2 Results Analysis and Discussions

Experiment results (for test data set) using all the three techniques are depicted in Table 5. As evident from Table 5, MEP obtained the best results for Normal, U2R and R2L types of attacks. LGP gave the best test results for Probe and DoS classes but did not perform well for U2R class. GEP performed moderately well for all the attack classes. In Table 6 the variable combinations evolved by MEP are presented. In Table 6, var represents Variable number (Column 1 in Table 1). It is to be noted that only these few variables are required to detect the a particular type of attack. This leads to a very light intrusion detection system when compared to a fuzzy expert system (which requires so many rules) or a artificial neural network with so many hidden neurons [1, 2].

We also analyzed the True Positive Rate (TPR) and False Positive Rates (FPR) for MEP and GEP in order

Table 1: Variables for intrusion detection data set

Variable No.	Variable name	Variable type	Variable label
1	duration	continuous	A
2	protocol_type	discrete	B
3	service	discrete	C
4	flag	discrete	D
5	src_bytes	continuous	E
6	dst_bytes	continuous	F
7	land	discrete	G
8	wrong_fragment	continuous	H
9	urgent	continuous	I
10	hot	continuous	J
11	num_failed_logins	continuous	K
12	logged_in	discrete	L
13	num_compromised	continuous	M
14	root_shell	continuous	N
15	su_attempted	continuous	O
16	num_root	continuous	P
17	num_file_creations	continuous	Q
18	num_shells	continuous	R
19	num_access_files	continuous	S
20	num_outbound_cmds	continuous	T
21	is_host_login	discrete	U
22	is_guest_login	discrete	V
23	count	continuous	W
24	srv_count	continuous	X
25	serror_rate	continuous	Y
26	srv_serror_rate	continuous	X
27	rerror_rate	continuous	AA
28	srv_rerror_rate	continuous	AB
29	same_srv_rate	continuous	AC
30	diff_srv_rate	continuous	AD
31	srv_diff_host_rate	continuous	AE
32	dst_host_count	continuous	AF
33	dst_host_srv_count	continuous	AG
34	dst_host_same_srv_rate	continuous	AH
35	dst_host_diff_srv_rate	continuous	AI
36	dst_host_same_src_port_rate	continuous	AJ
37	dst_host_srv_diff_host_rate	continuous	AK
38	dst_host_serror_rate	continuous	AL
39	dst_host_srv_serror_rate	continuous	AM
40	dst_host_rerror_rate	continuous	AN
41	dst_host_srv_rerror_rate	continuous	AO

Table 2: Parameter setting for LGP

Parameter	Value				
	Normal	Probe	DoS	U2R	R2L
Population size	2048	2048	2048	2048	2048
Maximum no of tournaments	120000	120000	120000	120000	120000
Tournament size	8	8	8	8	8
Mutation frequency (%)	85	82	75	86	85
Crossover frequency (%)	75	70	65	75	70
Number of demes	10	10	10	10	10
Maximum program size	256	256	256	256	256

Table 3: Parameters used by MEP

Parameter	Value				
	Normal	Probe	DoS	U2R	R2L
Population size	100	200	250	100	100
Number of generations	30	200	800	20	800
Chromosome length	30	40	40	30	40
Crossover frequency (%)	90	90	80	90	90
No. of mutations per chromosome	3	4	5	3	4

Table 4: Values of parameters used by GEP

Parameter	Value				
	Normal	Probe	DoS	U2R	R2L
Population size	100	100	100	100	100
Number of generations	800	500	500	500	500
Number of genes	12	12	12	14	12
Mutation	0.1	0.1	0.1	0.1	0.1
One point crossover	0.4	0.4	0.3	0.4	0.4
Two points crossover	0.4	0.4	0.3	0.4	0.4
Gene recombination	0.2	0.2	0.1	0.1	0.2
Gene transposition	0.2	0.2	0.1	0.1	0.2

Table 5: Performance comparison between LGP, MEP and MEP

	Classification Accuracy on Test Data Set (%)				
	Normal	Probe	DoS	U2R	R2L
LGP	99.73	**99.89**	**99.95**	64.00	99.47
MEP	**99.82**	95.39	98.94	**99.75**	**99.75**
GEP	99.80	97.84	95.64	99.64	98.92

Table 6: Functions evolved by MEP

Attack type	Evolved Function
Normal	$var12 * log_2(var10 + var3)$
Probe	$(fabs(var30 + var35)) < (var26 + var27)?(fabs(var30 + var35)) : (var26 + var27);$
DOS	$var38 - (Ln(var41 * var6) + sin(Lg(var30))) - (Lg(var30) - (var41 * var6)) > (0.3415 + var24 + var41 * var6)?(var38 - (Ln(var41 * var6) + sin(Lg(var30))) - (Lg(var30) - (var41 * var6))) : (0.3415 + var24 + var41 * var6) + var8$
U2R	$sin(var14) - var33$
R2L	$fabs((fabs(var8 > (var1 + (var6 > (Ln(var6))?var6 : (Ln(var6))) * var3)?var10 : (var1 + (var6 > (Ln(var6))?var6 : (Ln(var6))) * var3))) * (var12 + var6)) + var11$

to determine the efficiency of the developed IDP's [32]. The values of TPR and FPR obtained by MEP and GEP for the test data set are depicted in Table 7. The true positive rate (TP) is given by:

$$TP = \frac{\text{positives correctly classified}}{\text{total positives}}$$

The false positive rate (FP) is given by:

$$FP = \frac{\text{total negatives - negatives } in\text{correctly classified}}{\text{total negatives}}$$

As evident from Table 7, when compared to LGP, MEP obtained the best values for TPR and FPR for Normal and DoS and good performance for the other classes. MEP required only 30 and 20 generations for the Normal and U2R types respectively.

MEP performance is illustrated in Figures 1 and 2. The classification accuracy for the best results obtained for training data, average of results obtained for the test data using the best trained function and the best results obtained for the test data are depicted. Figure 1(a) corresponds to Normal, Figure 1(b) corresponds to Probe, Figure 1(c) corresponds to DOS, Figure 1(d) corresponds to U2R and Figure 1(e) corresponds to R2L class respectively.

In Figure 2, the average of classification accuracies for the results obtained for training data and for the results obtained for the test data are depicted. Figure 2(a) corresponds to Normal, Figure 2(b) corresponds to Probe, Figure 2(c) corresponds DOS, Figure 2(d) corresponds to U2R and Figure 2(e) corresponds to R2L class respectively.

Figures 3 - 7 illustrate the growth in the program codes for LGP during the 120 tournaments. The best and average code length is depicted during the evolutionary learning.

Most machine learning paradigms (artificial neural networks, support vector machines, decision trees etc.) examine all the input features to detect intrusions or misuse patterns. Some of the features may be redundant or

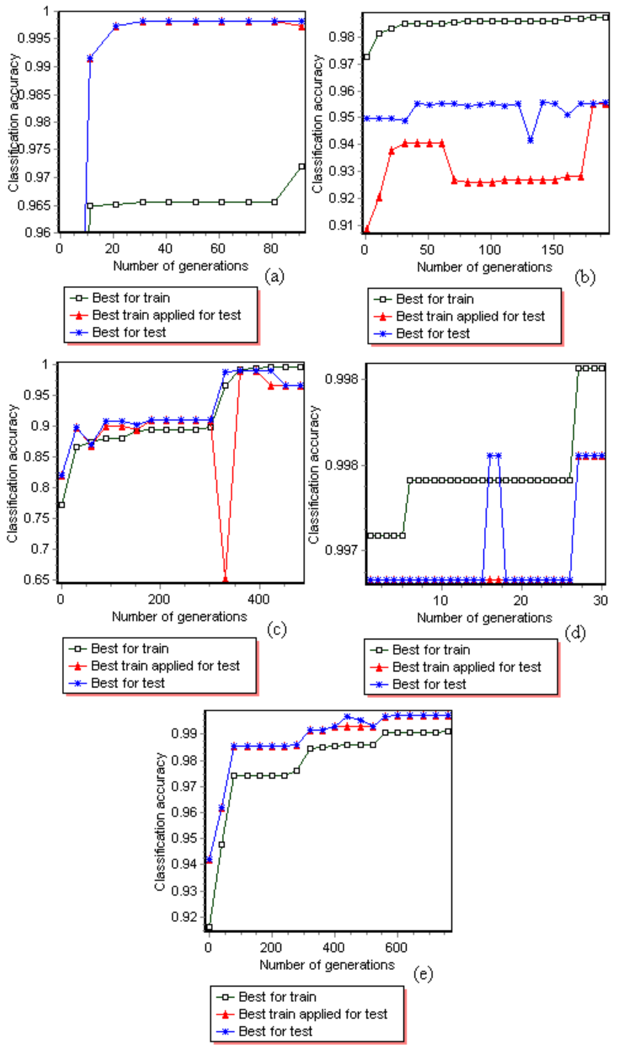

Figure 1: Relation between accuracy and number of generations

Table 7: Comparison of false alarm rates

Attack type	MEP				GEP	
	Best result for train applied for test		Best result for test			
	TP	FP	TP	FP	TP	FP
Normal	**0.996**	**0.999**	0.996	0.999	0.996	0.999
Probe	0.548	**0.999**	0.947	0.982	**0.999**	0.9748
DoS	0.986	0.995	**0.987**	**0.999**	0.918	0.943
U2Su	0.40	**0.999**	0.400	0.999	**0.437**	0.995
R2L	0.969	**1**	0.973	1	**0.989**	0.984

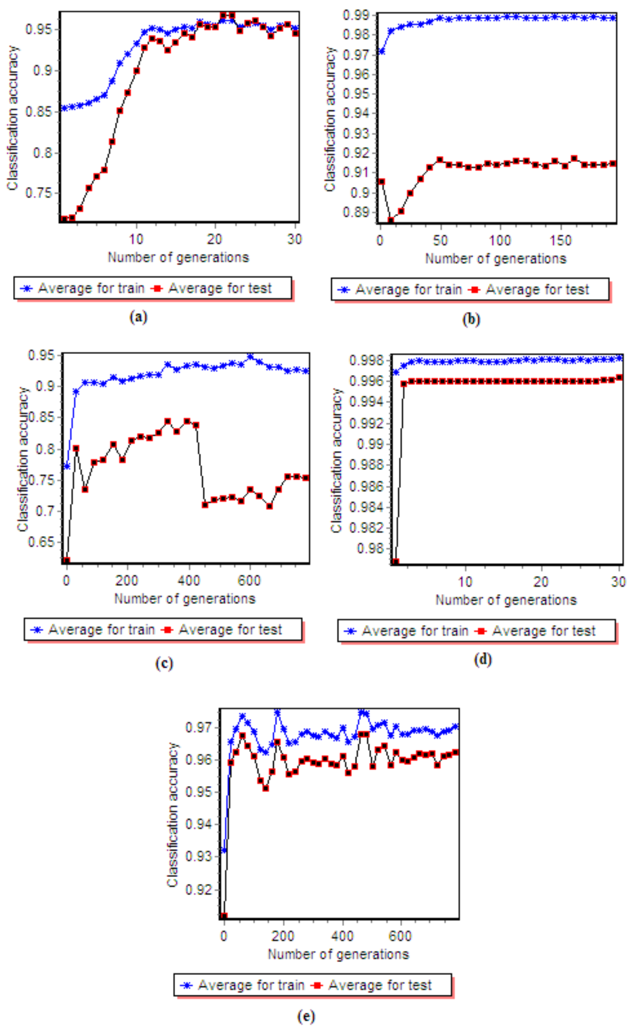

Figure 2: Average of classification accuracy for the results obtained by MEP for training and test data

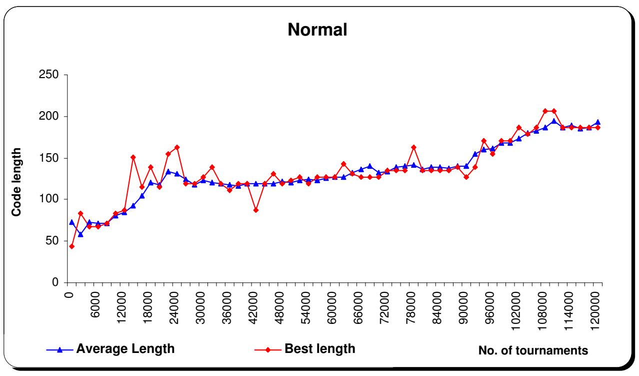

Figure 3: Growth of LGP program codes for normal class

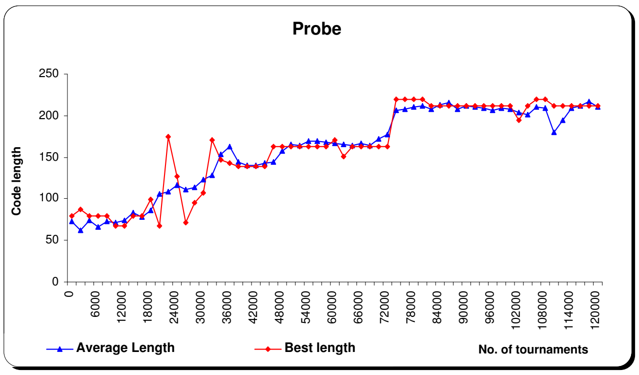

Figure 4: Growth of LGP program codes for probe class

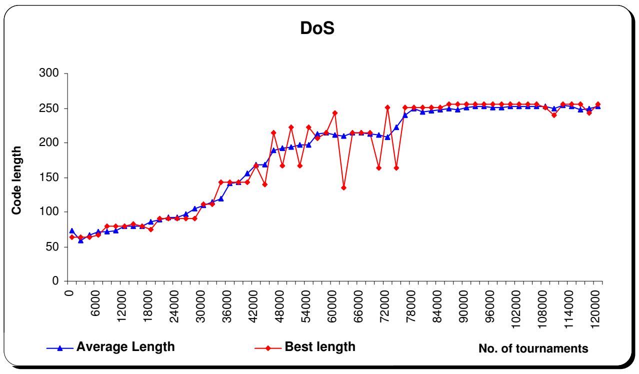

Figure 5: Growth of LGP program codes for DOS class

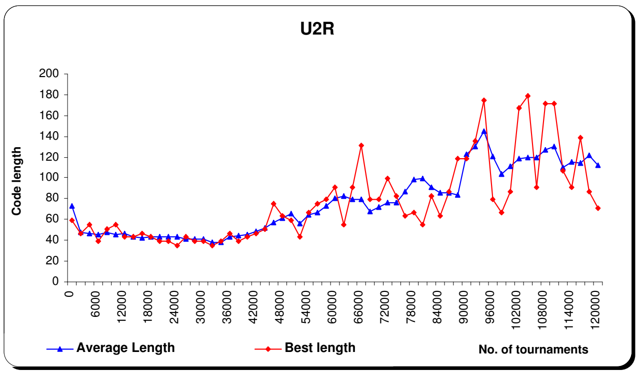

Figure 6: Growth of LGP program codes for U2R class

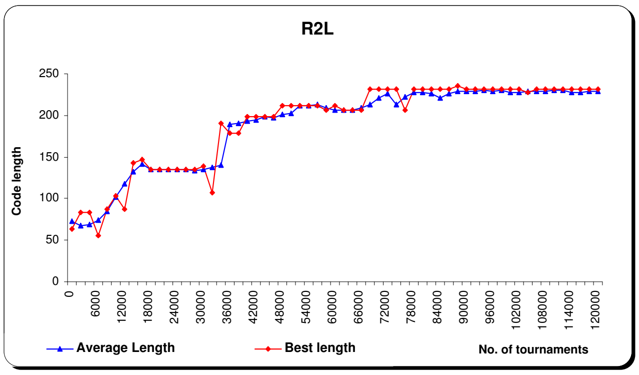

Figure 7: Growth of LGP program codes for R2L class

contribute little to the detection process (even could affect the performance adversely). The presented GP techniques could be used to pick up significant features randomly and explore the performance. As depicted in Table 6, MEP required only 3, 4, 6, 2 and 7 features (instead of the complete 41 features) to detect the Normal, Probe, DoS, U2R and R2L types of attacks respectively.

In some classes the accuracy figures tend to be very small and may not be statistically significant, especially in view of the fact that the 5 classes of patterns differ in their sizes tremendously. For example only 27 data sets were available for training the U2R class. More definitive conclusions can only be made after analyzing more comprehensive sets of network traffic.

5 Conclusions

This paper illustrated the importance of GP techniques for developing intrusion detection systems. MEP outperformed LGP for three of the considered attack classes and LGP outperformed MEP for two of the attack classes. GEP also obtained good results for all of the classes. MEP and GEP classification accuracy is grater than 95% for all considered classes. For three classes, MEP classification accuracy is greater than 99.75%. It is to be noted that for real time intrusion detection systems MEP, LGP and GEP would be the ideal candidates since they can be implemented at machine code level.

Perhaps the greatest advantage of genetic programming comes from the ability to develop IDP's for which there are no human experts. Although human expertise should be used when it is available, it often proves less than adequate for automating problem-solving routines.

References

[1] A. Abraham, "Evolutionary computation in intelligent web management", *Evolutionary Computing in Data Mining*, Ghosh A. and Jain L.C. (Eds.), Studies in Fuzziness and Soft Computing, Ch. 8, pp. 189-210, Springer Verlag, 2004.

[2] A. Abraham and J. Thomas, "Distributed intrusion detection systems: A computational intelligence approach, applications of information systems to homeland security and defense", Abbass H. A. and Essam D. (Eds.), Idea Group Inc. Publishers, USA, Ch. 5, pp. 105-135, 2005.

[3] J. P. Anderson, "Computer security threat monitoring and surveillance". *Technical report*, no. 79F296400, James P Anderson Co., Fort Washington, Pennsylvania, Apr. 1980.

[4] R. Beghdad, "Modelling and solving the intrusion detection problem in computer networks", *Computers & Security*, vol. 23, issue 8, pp. 687-696, 2004.

[5] M. Brameier and W. Banzhaf, "A comparison of linear genetic programming and neural networks in

medical data mining", IEEE Transactions on Evolutionary Computation, vol. 5, no. 1, pp. 17-26, 2001.

[6] M. Brameier and W. Banzhaf, "Explicit control of diversity and effective variation distance in Linear Genetic Programming", in *Proceedings of the fourth European Conference on Genetic Programming*, LNCS 2278, Springer-Verlag, 2002.

[7] L. Brieman, J. Friedman, R. Olshen, and C. Stone, "Classification of regression trees", Wadsworth Inc., 1984.

[8] A. Boukerche, K. R. Lemos Juc, J. B. Sobral, and M. Sechi Moretti Annoni Notare, "An artificial immune based intrusion detection model for computer and telecommunication systems", *Parallel Computing*, vol. 30, issues 5-6, pp. 629-646, 2004.

[9] S. Chebrolu, A. Abraham, and J. P. homas, "Feature deduction and ensemble design of intrusion detection systems", *Computers & Security*, vol. 24, issue 4, pp. 295-307, 2005.

[10] W. H. Chen, S. H. Hsu, and H. P. Shen, "Application of SVM and ANN for intrusion detection", *Computers & Operations Research*, vol. 32, issue 10, pp. 2617-2634, 2005.

[11] Y. Chen, A. Abraham, and J. Yang, "Feature deduction and intrusion detection using flexible neural trees", in *Second IEEE International Symposium on Neural Networks (ISNN 2005)*, LNCS 3498, pp. 439-446, Springer-Verlag, 2005.

[12] D. Dasgupta, F. Gonzalez, K. Yallapu, J. Gomez, and R. Yarramsettii, "CIDS: An agent-based intrusion detection system", *Computers & Security*, vol. 24, issue 5, pp. 387-398, 2005.

[13] D. Denning, "An intrusion-detection model", *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, 1987.

[14] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks", *Expert Systems with Applications*, vol. 29, issue 4, pp. 713-722, 2005.

[15] C. Ferreira, "Gene Expression Programming: A new adaptive algorithm for solving problems". *Complex Systems*, vol. 13, issue 2, pp. 87-129, 2001.

[16] C. Ferreira, "Genetic representation and genetic neutrality in gene expression programming". *Advances in Complex Systems*, vol. 5 no. 4, pp. 389-408, 2002.

[17] J. H. Friedman, "Multivariate adaptative regression splines", *Annals of Statistics,* vol. 19, 1991.

[18] T. D. Garvey and T. F. Lunt, "Model based intrusion detection", in *Proceedings of the 14th National Computer Security Conference*, pp. 372-385, Oct. 1991.

[19] C. Grosan, A. Abraham, and S. Y .Han, "MEPIDS: Multi-expression programming for intrusion detection system", in *International Work-conference on the Interplay between Natural and Artificial Computation, (IWINAC'05)*, LNCS 3562, pp. 163-172, Springer-Verlag, 2005.

[20] R. Heady, G. Luger, A. Maccabe, and M. Servilla, "The architecture of a network level intrusion detection system". *Technical report*, Department of Computer Science, University of New Mexico, 1990.

[21] K. Ilgun, "USTAT: A real-time intrusion detection system for UNIX, master thesis", University of California, Santa Barbara, 1992.

[22] W. Jiang, H. Song, and Y. Dai, "Real-time intrusion detection for high-speed networks", *Computers & Security*, vol. 24, issue 4, pp. 287-294, 2005.

[23] KDD Cup 1999 Intrusion detection data set: http://kdd.ics.uci.edu/databases/kddcup99 /kddcup.data_10_percent.gz

[24] S. Kumar and E. H. Spafford, "An application of pattern matching in intrusion detection". *Technical Report CSD-TR-94-013*, Purdue University, 1994.

[25] W. Lee, S. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models", in *proceedings of the IEEE Symposium on Security and Privacy*, pp. 120-132, 1999.

[26] MIT Lincoln Laboratory. http://www.ll.mit.edu/IST/ideval/

[27] S. Mukkamala, A. Sung, and A. Abraham, "Intrusion detection using ensemble of soft computing and hard computing paradigms", *Journal of Network and Computer Applications, Elsevier Science*, vol. 28, issue 2, pp. 167-182, 2005.

[28] M. Oltean and C. Grosan, "A cComparison of several linear GP techniques, *Complex Systems*, vol. 14, no. 4, pp. 285-313, 2004.

[29] M. Oltean and C. Grosan, "Evolving evolutionary algorithms using multi expression programming. in *Proceedings of The* 7^{th} *European Conference on Artificial Life*, Dortmund, Germany, pp. 651-658, 2003.

[30] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems", *Journal of Network and Computer Applications*, 2006 (in press).

[31] P. A. Porras, "STAT: A state transition analysis tool for intrusion detection", Master's Thesis, Computer Science Dept., University of California, Santa Barbara, 1992.

[32] F. Provost and T. Fawcett. "Robust classification for imprecise environment"s, Machine Learning 42, pp. 203-231, 2001.

[33] V. Ramos and A. Abraham, "ANTIDS: Self organized ant based clustering model for intrusion detection system", in *The Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05)*, Japan, pp. 977-986, Springer-Verlag, 2005.

[34] G. T. Rohrmair and G. Lowe, "Using data-independence in the analysis of intrusion detection systems", *Theoretical Computer Science*, vol. 340, issue 1, pp. 82-101, 2005.

[35] C. Ryan, J. J. Collins, and M. O'Neill, "Gramatical evolution: Evolving programs for an arbitrary language", in *proceedings of the first European Work-*

shop on Genetic Programming, pp. 83-96, Springer-Verlag, 1998.

[36] S. L. Scott, "A bayesian paradigm for designing intrusion detection systems", *Computational Statistics & Data Analysis*, vol. 45, issue 1, pp. 69-83, 2004.

[37] R. C. Summers, *Secure computing: Threats and safeguards*. McGraw Hill, New York, 1997.

[38] A. Sundaram, "An introduction to intrusion detection". *ACM Cross Roads*, vol. 2, no. 4, Apr. 1996.

[39] H. S. Teng, K. Chen, and S. C. Lu, "Security audit trail analysis using inductively generated predictive rules", in *proceedings of the 11th National Conference on Artificial Intelligence Applications*, Piscataway, pp. 24-29, Mar. 1990.

[40] V. N. Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag, 1995.

[41] C. Zhang, J. Jiang, and M. Kamel, "Intrusion detection using hierarchical neural networks", *Pattern Recognition Letters*, vol. 26, issue 6, pp. 779-791, 2005.

[42] Z. Zhang, and H. Shen, "Application of online-training SVMs for real-time intrusion detection with different considerations", *Computer Communications*, vol. 28, issue 12, pp. 1428-1442, 2005.

Ajith Abraham currently works as a Professor under the South Korean Government's Institute of Information Technology Assessment (IITA) Professorship program at Chung-Ang University, Korea and is a visiting researcher at Rovira i Virgili University, Tarragona, Spain. His primary research interests are in computational intelligence with a focus on using evolutionary computation techniques for designing intelligent paradigms. Application areas include Web services, information security, Web intelligence, financial modeling, multi criteria decision-making, data mining etc. He has authored/co-authored over 200 research publications in peer reviewed reputed journals, book chapters and conference proceedings of which three have won 'best paper' awards. He is serving the Editorial board of over a dozen International Journals and has also guest edited 15 special issues for reputed International Journals. He received PhD degree from Monash University, Australia. More information at: http://www.softcomputing.net

Crina Grosan currently works as an Assistant Professor in the Computer Science Department of Babes-Bolyai University, Cluj-Napoca, Romania. She received her PhD degree from Babes-Bolyai University, Romania. Her main research area is in Evolutionary Computation, with a focus on Evolutionary Multiobjective Optimization and applications and Genetic Programming. She is also interested in Swarm Intelligence (Particle Swarm Optimization, Ant Colonies Systems), Bioinformatics, Financial Modeling, etc. Crina Grosan authored/co-authored over 60 papers in peer reviewed international journals, proceedings of the international conferences and book chapters. She is co-author of two books in the field of computer science. She is Managing Editor of the International Journal of Computational Intelligence Research (IJCIR). Dr. Grosan is the co-editor for the following books which will be published by Springer Verlag, Germany: Swarm Intelligence in Data Mining, Stigmergic Optimization, Hybrid Evolutionary Systems.

Carlos Martin-Vide is, since 1992, Professor and Head of the Research Group on Mathematical Linguistics at Rovira i Virgili University, Tarragona, Spain. His areas of expertise are formal language theory, mathematical linguistics, theoretical computer science and biomolecular computing. His volumes recently edited are: Where Mathematics, Computer Science, Linguistics and Biology Meet (Kluwer, 2001, with V. Mitrana), Grammars and Automata for String Processing: From Mathematics and Computer Science to Biology, and Back (Taylor and Francis, 2003, with V. Mitrana), Membrane Computing (LNCS 2933, Springer, 2004, with G. Mauri, G. Paun, G. Rozenberg and A. Salomaa), Formal Languages and Applications (Springer, 2004, with V. Mitrana and G. Paun), and Recent Advances in Formal Languages and Applications (Springer, 2006, with Z. ?sik and V. Mitrana). He is (co-)author of more than 255 papers in peer-reviewed journals and conference proceedings. Journals where his work was published include: Acta Cybernetica (2); Acta Informatica (4); BioSystems (2); Computational Linguistics; Computers and Artificial Intelligence; Discrete Applied Mathematics; Electronic Notes in Theoretical Computer Science (2); Fundamenta Informaticae (6); Information Processing Letters; Information Sciences; International Journal of Computer Mathematics; International Journal of Foundations of Computer Science (2); International Journal of Pattern Recognition and Artificial Intelligence; Journal of Automata, Languages and Combinatorics (5); Journal of Parallel and Distributed Computing; Journal of Universal Computer Science (3); Linguistics and Philosophy; Natural Computing (2); New Generation Computing; Publicationes Mathematicae Debrecen (2); Theoretical Computer Science (7). He published 28 papers in LNCS volumes. He is the editor-in-chief of the journal Grammars (1998-), the chairman of the International PhD School in Formal Languages and Applications (Tarragona, 2001-), and the Chairman of the International PhD School in Language and Speech Technologies (Tarragona, 2005-).