Evolution of Intrusion Detection Systems

Ajith Abraham, Crina Grosan[1], Yuehui Chen[2]

School of Computer Science and Engineering, Chung-Ang University, Korea
ajith.abraham@ieee.org

[1]Department of Computer Science
Babes-Bolyai University, Cluj-Napoca, 3400, Romania
cgrosan@cs.ubbcluj.ro

[2]School of Information Science and Engineering
Jinan University, Jinan 250022, P.R.China
yhchen@ujn.edu.cn

Abstract. Several information security techniques are available today to protect information systems against unauthorized use, duplication, alteration, destruction and virus attacks. An Intrusion Detection System (IDS) is a program that analyzes what happens or has happened during an execution and tries to find indications that the computer has been misused. This article presents some of the challenges in designing efficient intrusion detection systems which could provide high accuracy, low false alarm rate and reduced number of features. Finally, we present how some of the computational intelligence paradigms could be used in designing intrusion detection systems in a distributed environment.

1 . Intrusion Detection Systems

Attacks on the computer infrastructures are becoming an increasingly serious problem. Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability. 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. Computing systems refer to computers, computer networks, and the information they handle. 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. An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. 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.

Intrusion detection is classified into two types: misuse and anomaly detection. Misuse intrusion detection uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions. These patterns are encoded in advance and used to match against the user behavior to detect intrusion. Anomaly intrusion detection uses the normal usage behavior patterns to identify the intrusion. The normal usage patterns are constructed from the statistical measures of the system features. The behavior of the user is observed and any deviation from the constructed normal behavior is detected as intrusion.

Dorothy Denning [1] 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. Different techniques and approaches have been used in later developments. Some of the techniques used are statistical approaches, predictive pattern generation, expert systems, keystroke monitoring, state transition analysis, pattern matching, and data mining techniques. Figure 1 illustrates a simple network, which is protected using IDS.

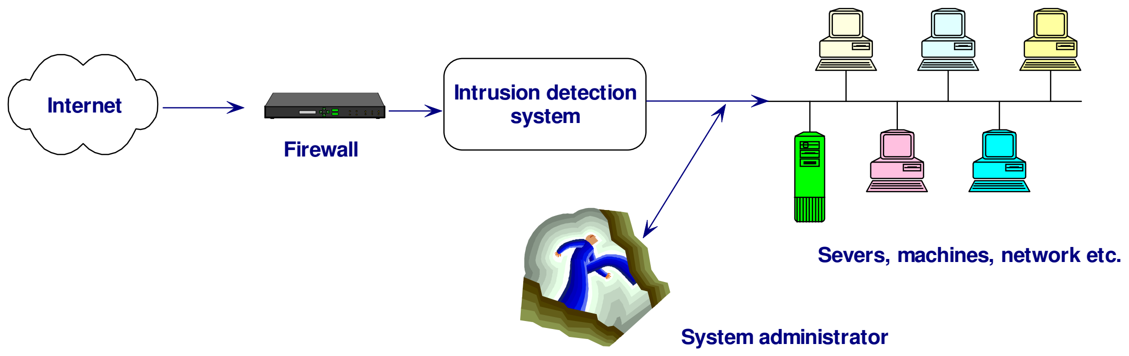

Figure 1. Network protection using conventional IDS

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. Any user behavior that deviates significantly from this normal behavior is flagged as an intrusion. Predictive pattern generation uses a rule base of user profiles defined as statistically weighted event sequences. This method of intrusion detection attempts to predict future events based on events that have already occurred. State transition analysis approach uses the state transitions of the system to identify the intrusions. State transition diagrams list only the critical events that must occur for the successful completion of the intrusion. Keystroke monitoring technique utilizes user's keystrokes by a pattern matching of the sequence of keystrokes to some predefined sequences to detect the intrusion. The main problems with this approach are lack of support from operating system to capture the keystroke sequences and also many ways of expressing the sequence of keystrokes for same attack.

Expert systems have played an important role to build IDS. The rules may recognize single auditable events that represent significant danger to the system by themselves, or they may recognize a sequence of events that represent an entire penetration scenario.

2. Computational Intelligence Paradigms

Computational intelligence approaches for intrusion detection was first implemented in mining audit data for automated models for intrusion detection [3]. Raw data is converted into ASCII network packet information, which in turn is converted into connection level information. These connection level records contain within connection features like service, duration etc. Besides several machine learning techniques and artificial immune systems, several intelligent paradigms have been explored to create models to detect intrusions.

1. Artificial neural networks (ANN) have been used both in anomaly intrusion detection as well as in misuse intrusion detection. The Bayesian Neural Network (BNN) is a powerful knowledge representation and reasoning algorithm under conditions of uncertainty [7]. A Bayesian network B = (N, A, Θ) is a Directed Acyclic Graph (DAG) (*N, A*) where each node $n \in N$ represents a domain variable (e.g. a dataset attribute or variable), and each arc $a \in$ A between nodes represents a probabilistic dependency among the variables, quantified using a conditional probability distribution (CP table) $\theta_i \in \Theta$ for each node n_i.
2. Support vector machines (SVM) have proven to be a good candidate for intrusion detection because of its training speed and scalability [6].
3. Several variants of evolutionary algorithms have been used for designing intrusion detection systems. Linear Genetic Programming (LGP) is a variant of the conventional Genetic Programming (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++) [8] . In Multi Expression Programming (MEP) a chromosome encodes more than one problem solution. The chromosome fitness is usually defined as the fitness of the best expression encoded by that chromosome.
4. Fuzzy logic has proved to be a powerful tool for decision making to handle and manipulate imprecise and noisy data. Two different types of fuzzy classifiers have been used. The first classifier (FR_1) 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 (FR_2). The third method uses a neuro-fuzzy computing framework in which a fuzzy inference system is learned using neural network learning

paradigms [8]. Readers are advised to refer to the author's previous work for detailed information regarding the generation of fuzzy if-then rules [8].

5. Multivariate Adaptive Regression Splines (MARS) is an innovative approach that automates the building of accurate predictive models for continuous and binary dependent variables [5]. It excels at finding optimal variable transformations and interactions, and the complex data structure that often hides in high-dimensional data.
6. 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 [10].
7. 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 [2][7].
8. Several hybrid approaches for modeling IDS have been also explored. Decision Trees (DT) and Support Vector Machines (SVM) are combined as a hierarchical hybrid intelligent system model (DT-SVM) [8].
9. An IDS based on general and enhanced Flexible Neural Tree (FNT) is explored by Chen and Abraham [9]. 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.

Sections 3 and 4 illustrates the empirical results obtained by using the above paradigms for designing intrusion detection systems.

3.0 How Accurate IDS could be?

The accuracy of the computational intelligent paradigms was verified by some simulations using the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Labs [4]. The LAN was operated like a real environment, but was blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. The 41 features are labeled in order as *A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, AA, AB, AC, AD, AF, AG, AH, AI, AJ, AK, AL, AM, AN, AO* and the class label is named as *AP*. 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.

U2Su: 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.

The designing of IDS involves training and testing phase. In the training phase, the different computational intelligence paradigms 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. This data set has five different classes namely *Normal, Probes, DoS, R2L* and *U2R.* A 5-class classification is performed to validate the accuracy of IDS. The detection accuracies (for test data set) for all the 13 different computational intelligence paradigms (presented in Section 2) are depicted in Table 1. For MEP (as a sample), the false alarm rates are presented in Table 2. The indices for a particular class of attack are determined as follows: True positive rate = (number of attack patterns correctly classified/total number of attack patterns). False positive rate = (number of non-attack patterns correctly classified/total number of non-attack patterns).

Table 1. Performance comparison using full data set[1]

	Classification accuracies for different attack types				
	Normal	**Probe**	**DoS**	**U2R**	**R2L**
ANN	99.60	92.70	97.50	48	95.00
BNN	99.57	99.43	99.69	64	99.11
SVM	99.64	98.57	99.92	40.00	33.92
DT	99.64	99.86	96.83	68.00	84.19
LGP	99.73	99.89	99.95	64.00	99.47
MEP	99.82	95.39	98.94	99.75	99.75
FR_1	40.44	53.06	60.99	66.75	61.10
FR_2	100.00	100.00	100.00	100.00	100.00
FR_3	98.26	99.21	98.18	61.58	95.46
MARS	96.08	92.32	94.73	99.71	99.48
ANTIDS	99.94	98.29	99.97	64.00	99.47
DT-SVM	99.70	98.57	99.92	48.00	37.80
FNT	99.19	98.39	98.75	99.70	99.09

Table 2. False alarm rates

	False Alarm Rates				
	Normal	**Probe**	**DoS**	**U2R**	R2L
True +	0.9957	0.9471	0.9871	0.4000	0.9733
False +	0.9989	0.982	0.9992	0.9997	1.0000

4. Can Computational Intelligence Techniques Reduce Features?

Most of the existing 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. Since the amount of audit data that an IDS needs to examine is very large even for a small network. Analysis is difficult

[1] Readers may please refer to the references for more detailed results

even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns. Complex relationships exist between the features, which are practically impossible for humans to discover. An IDS must therefore reduce the amount of data to be processed. This is extremely important if real-time detection is desired. Reduction can occur in one of several ways. Data that is not considered useful can be filtered, leaving only the potentially interesting data. Data can be grouped or clustered to reveal hidden patterns. By storing the characteristics of the clusters instead of the individual data, overhead can be significantly reduced. Finally, some data sources can be eliminated using feature selection. Features may contain false correlations, which hinder the process of detecting intrusions. Further, some features may be redundant since the information they add is contained in other features. Extra features can increase computation time, and can impact the accuracy of IDS. Feature selection improves classification by searching for the subset of features, which best classifies the training data. The features under consideration depend on the type of IDS, for example, a network based IDS will analyze network related information such as packet destination IP address, logged in time of a user, type of protocol, duration of connection etc. It is not known which of these features are redundant or irrelevant for IDS and which ones are relevant or essential for IDS. There does not exist any model or function that captures the relationship between different features or between the different attacks and features. If such a model did exist, the intrusion detection process would be simple and straightforward. In this paper we use data mining techniques for feather selection. The subset of selected features is then used to detect intrusions.

Markov Blanket Modeling of Input Features

Markov Blanket (MB) of the output variable *T,* is a novel idea for significant feature selection in large data sets. *MB* (*T*) is defined as the set of input variables such that all other variables are probabilistically independent of *T*. A general BN classifier learning is that we can get a set of features that are on the Markov blanket of the class node. The Markov blanket of a node *n* is the union of *n*'s parents, *n*'s children and the parents of *n*'s children. The formal definition is:

The Markov blanket of a feature *T*, *MB*(*T*) of a BN. the set of parents, children, and parents of children of *T*. *MB*(*T*) is the minimal set of features conditioned on which all other features are independent of *T*, i.e. for

any feature set *S*, $P(T \mid MB(T), S) = P(T \mid MB(T))$.

Knowledge of *MB*(*T*) is sufficient for perfectly estimating the distribution of *T* and thus for classifying *T*. In order to solve the feature selection problem, one could induce the BN that generates the data. This subset of nodes shields *n* from being affected by any node outside the blanket. When using a BN classifier on complete data, the Markov blanket of the class node forms feature selection and all features outside the Markov blanket are deleted from the BN [7].

Decision Tree Learning and Feature Deduction

Feature selection is done based on the contribution the input variables make to the construction of the decision tree. Feature importance is determined by the role of each input variable either as a main splitter or as a surrogate. Surrogate splitters are defined as back-up rules that closely mimic the action of primary splitting rules. Suppose that, in a given model, the algorithm splits data according to variable '*protocol_type*' and if a value for '*protocol_type*' is not available, the algorithm might substitute *'service'* as a good surrogate. Variable importance, for a particular variable is the sum across all nodes in the tree of the improvement scores that the predictor has when it acts as a primary or surrogate (but not competitor) splitter. Example, for node *i*, if the predictor appears as the primary splitter then its contribution towards importance could be given as $i_{importance}$. But if the variable appears as the n^{th} surrogate instead of the primary variable, then the importance becomes $i_{importance} = (p^n) * i_{improvement}$ in which *p* is the 'surrogate improvement weight' which is a user controlled parameter set between (0-1) [7].

Felxible Neural Trees (FNT)

The mechanisms of input selection in the FNT constructing procedure are as follows. (1) Initially the input variables are selected to formulate the FNT model with same probabilities; (2) The variables which have more contribution to the objective function will be enhanced and have high opportunity to survive in the next generation by a evolutionary procedure; (3) The evolutionary operators i.e., crossover and mutation, provide a input selection method by which the FNT should select appropriate variables automatically [9].

4.1. Feature Deduction Experiments

A comparison between three different models for feature deduction using the same datasets mentioned in the previous section. Markov blanket model algorithm helps to reduce the 41 variables to 17 variables. These 17 variables are *A, B, C, E, G, H, K, L, N, Q, V, W, X, Y, Z, AD* and *AF*.

The important variables were also decided by their contribution to the construction of the decision tree. Variable rankings were generated in terms of percentages. We eliminated the variables that had 0.00% rankings and considered only the primary splitters or surrogates. This resulted in a reduced 12 variable data set with *C, E, F, L, W, X, Y, AB, AE, AF, AG* and *AI* as variables.

FNT method helps to reduce the features as given below.
Normal*:* *C, K, U, AN*
Probe: *A, C, L, R, T, U, W, Z, AA, AE, AK, AO*
DoS: *A, H, J, K, P, Q, T, U, W, AB, AB, AC, AE*
U2R: *K, N, Q, AB, AC, AF, AJ, AL*
R2L: *A, C, K, L, M, R, T, W, Y, AL*

Table 3 provides a comparison between the three feature reduction methods for the test data set for detecting the different attacks.

Table 3. Performance comparison using reduced dataset[2]

Attack type	**17 variables**	**12 variables**	**4~13 variables**
	BNN	**DT**	**FNT**
Normal	99.64	**100.00**	99.19
Probe	**98.57**	97.71	98.39
DOS	98.16	85.34	**98.75**
U2R	60.00	64.00	**99.70**
R2L	98.93	95.56	**99.09**

[2] Readers may please refer to the references for more detailed results

5. Distributed Intrusion Detection System (DIDS)

In Distributed IDS (DIDS) conventional intrusion detection system are embedded inside intelligent agents and are deployed over a large network. In a distributed environment, IDS agents communicate with each other, or with a central server. Distributed monitoring allows early detection of planned and coordinated attacks and thereby allowing the network administrators to take preventive measures. DIDS also helps to control the spreading of worms, improves network monitoring and incident analysis, attack tracing and so on. It also helps to detect new threats from unauthorized users, back-door attackers and hackers to the network across multiple locations, which are geographically separated. In a DIDS it is important to ensure that the individual IDS are light-weight and accurate.

A number of IDS have been proposed for a networked or distributed environment. Software agents have been proposed as a technology for intrusion detection applications. Rationale for considering agents in an IDS ranges from increased adaptability for new threats to reduced communication costs. Since agents are independently executing entities, there is the potential that new detection capabilities can be added without completely halting, rebuilding, and restarting the IDS. The autonomous agents have been used for data collection and analysis. Agents hosted on network nodes act as filters to extract pertinent data, transceivers to oversee agent operation, and monitors to receive reports from transceivers. These entities are organized into a hierarchical architecture with centralized control. Cooperating Security Managers (CSM) enable individual distributed intrusion detection packages to cooperate in performing network intrusion detection without relying on centralized control. Each individual CSM detects malicious activity on the local host. When suspicious activity is detected, each CSM will report any noteworthy activity to the CSM on the host from which the connection originated. The local CSM will not notify all networked systems, but rather only the system immediately before it in the connection chain. DIDS are simply a superset of the conventional IDS implemented in a distributed environment. Due to the distributed nature the implementation poses several challenges. IDS could be embedded inside agents and placed in the network to be monitored. The individual IDS may be configured to detect a single attack, or they may detect several types of attacks. Each network component may host one or many IDS. Since there will be a large number of flag generators (detection of an attack, event etc.), these must be abstracted, analyzed, and condensed by a suitable architecture before arriving a final conclusion. Very often there would be a centralized analyzing and control facility. The most popular architecture is the master-slave type which may be suitable for small networks. In a hierarchical architecture analysis and control are being done at different layers mainly because of the geographical distribution or due to the size of the network. Attacks/event detection information is passed to analyzer/controller nodes that aggregate information from multiple IDS agents. It is to be noted that the event information, which is detected by the IDS agents will follow a bottom up approach for analysis and the various command and control flow will follow a top-down approach. The physical location of IDS agents will be fixed since they monitor fixed network segments. In the case of hierarchical architecture, the analyzer/controller nodes may exist at many locations in the network since they receive their input and give their output via network connections. Depending on the network environment the communication between the different layers could be implemented as depicted in Figure 2.

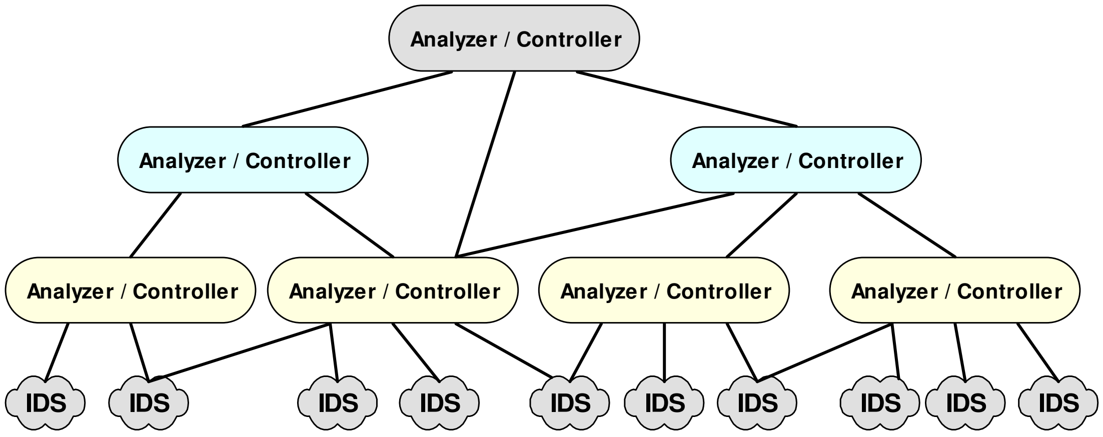

Figure 2. Hierarchical architecture with free communication between layers.

In the hierarchical architecture, the Central Analyzer and Controller (CAC) is the heart and soul of the DIDS. The CAC usually consists of a database and Web server, which allows interactive querying by the network administrators for attack information/analysis and initiate precautionary measures. CAC also performs attack aggregation, building statistics, identify attack patterns and perform rudimentary incident analysis. The co-operative intelligent agent network is one of the most important components of the DIDS. Ideally these agents will be located on separate network segments, and very often geographically separated. Communication among the agents is done utilizing the TCP/IP sockets.

Agent modules running on the host machines are capable of data analysis and to formulate adequate response action and are very often implemented as read only and fragile. In the event of tampering or modification the agent reports to the server agent and automatically ends its life. Agents residing in the individual analyzer/controllers consist of modules responsible for agent regeneration, dispatch, updating and maintaining intrusion signatures and so on. These agents control the individual IDS agents for monitoring the network, manage all the communication and life cycle of the IDS agents and also updates the IDS agents with detection algorithms, response and trace mechanisms.

6. Discussions

Effective intrusion detection and management systems are critical components of homeland security as they are in the forefront of the battle again cyber-terrorism. This article was focused on three different perspectives in designing intrusion detection systems. (1) accuracy and false alarm rates (2) feature deduction and (3) implementation in a distributed environment. For real time intrusion detection systems, MEP/LGP would be the ideal candidate as it can be manipulated at the machine code level. These code based IDS not only provides high accuracy but also is light weight and we can easily implemented in a mobile agent environment which makes then ideal candidates in mobile/MANET environments. The functions developed for detecting the different attacks types using MEP is given below.

Attack	**Developed function**
Normal	L * log_2(J + C)
Probe	(log_2(B) < (*fabs*((AJ *AA) > (AA + AI { AH) ? (AI * AA): (AA + AI {AH)))? (log_2(B)) : (*fabs*((AJ * AA) > (AA + AI {AH) ? (AJ * AA): (AA + AI {AH)))
DOS	0.457 * (H + ($\ln$(F)) * ($\lg$(AO)) - - AN + W + H)

U2R	$sin(N)$ - - AG
R2L	0.36 + (K< 0.086? K: 0.086 + 0.086) > (F > (log_2(log_2(L * C))) ? F : (log_2(log_2 (L * C)))) ? (K< 0.086 ? K: 0.086 + 0.086) : (F > (log_2(log_2(L * C))) ? F: (log_2(log_2(M*C)))) + F

Overall, the fuzzy classifier (FR_2) gave 100% accuracy for all attack types using all the 41 attributes. These type of IDS could be useful for conventional static networks, wireless base stations etc.

8.0. Conclusions

With the increasing incidents of cyber attacks, building an effective intrusion detection model with good accuracy and real-time performance are essential. This field is developing continuously. In this article, we presented some of the computational intelligence paradigms which could be useful for designing accurate intrusion detection systems which could be also deployed in a distributed environment.

References

[1] Denning D., An Intrusion-Detection Model, IEEE Transactions on Software Engineering, Vol. SE-13, No. 2, pp.222-232, 1987.

[2] Brieman L., Friedman J., Olshen R., and Stone C., Classification of Regression Trees. Wadsworth Inc., 1984.

[3] Lee W. and Stolfo S. and Mok K., A Data Mining Framework for Building Intrusion Detection Models. In Proceedings of the IEEE Symposium on Security and Privacy, 1999.

[4] MIT Lincoln Laboratory. <http://www.ll.mit.edu/IST/ideval/>

[5] J H Friedman, Multivariate Adaptative Regression Splines. Annals of Statistics Vol. 19 1991

[6] Mukkamala S., Sung A. and Abraham A., 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.

[7] Chebrolu S., Abraham A. and Thomas J., Feature Deduction and Ensemble Design of Intrusion Detection Systems, Computers and Security, Elsevier Science, 2005. http://dx.doi.org/10.1016/j.cose.2004.09.008

[8] Abraham A. and Thomas J., 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, 2005. http://falklands.globat.com/~softcomputing.net/hussein_chapter.pdf

[9] Chen Y. and Abraham A., Feature Deduction and Intrusion Detection Using Flexible Neural Trees, Second IEEE International Symposium on Neural Networks (ISNN 2005), Lecture Notes in Computer Science, Springer Verlag, Germany 2005 (in press). http://falklands.globat.com/~softcomputing.net/issn05.pdf

[10] Vitorino Ramos and Ajith Abraham, ANTIDS: Self Organized Ant Based Clustering Model for Intrusion Detection System, The Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05), Muroran, Japan, 2005 (in press). http://falklands.globat.com/~softcomputing.net/wstst-ra.pdf

Author Biographies

Ajith Abraham currently works as a Distinguished Visiting Professor under the South Korean Government's Institute of Information Technology Assessment (IITA) Professorship programme at Chung-Ang University, Korea. His primary research interests are in computational intelligence with a focus on using evolutionary computation techniques for designing intelligent paradigms. Application areas include several real world knowledge-mining applications like information security, bioinformatics, Web intelligence, energy management, financial modelling, weather analysis, fault monitoring, multi criteria decision-making etc. He has associated with over 150 research publications in peer reviewed reputed journals, book chapters and conference proceedings of which three have won 'best paper' awards.

He is the founding Co Editor-in-Chief of The International Journal of Hybrid Intelligent Systems (IJHIS), IOS Press, Netherlands and currently serves the editorial board of seven other international journals. Since 2001, he is actively involved in the Hybrid Intelligent Systems (HIS) and the Intelligent Systems Design and Applications (ISDA) series of annual International conferences. He is also the General Co-Chair of the The Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST05), Muroran, Japan and the Program Co-Chair of the Inaugural IEEE Conference on Next Generation Web Services Practices, Seoul, Korea. During the last four years, he has also served the technical committee of over 40 AI related International conferences and has also given a number of conference tutorials in Europe and USA. He is a member of IEEE, IEEE (CS), ACM, IEE (UK), IEAust and also works closely with several academic working groups like EvoNet, EUSFLAT, WFSC, etc. He received PhD degree from Monash University, Australia. More information at: http://ajith.softcomputing.net

Crina Grosan currently works as an Assistant Professor in the Computer Science Department of Babes-Bolyai University, Cluj-Napoca, Romania. Her main research area is in Evolutionary Computation, with a focus on Evolutionary Multiobjective Optimization and applications and Genetic Programming. Crina Grosan authored/co-authored over 50 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 proposed few Evolutionary techniques for single and multiobjective optimization, a genetic programming technique for solving symbolic regression problems and so on. Crina Grosan is member of the IEEE (CS), IEEE (NN) and ISGEG. She has successfully completed the thesis defense and is expected to receive the PhD degree from Babes-Bolyai University, Romania during this summer.

Yuehui Chen received his B.Sc. degree in mathematics/automatics from the Shandong University of China in 1985, and Ph.D. degree in electrical engineering from the Kumamoto University of Japan in 2001. During 2001–2003, he had worked as the Senior Researcher of the Memory-Tech Corporation at Tokyo. Since 2003 he has been a member at the Faculty of Electrical Engineering in Jinan University, where he is currently head of the Laboratory of Computational Intelligence. His research interests include evolutionary computation, neural networks, fuzzy systems, hybrid computational intelligence and their applications in time-series prediction, system identification and intelligent control. He is the author and co-author of more than 60 papers. Prof. Chen is a member of IEEE, the IEEE Systems, Man and Cybernetics Society and the Computational Intelligence Society.