The Multi-agent Systems for Computer Network Security Assurance: Frameworks and Case Studies*

V. Gorodetski, I. Kotenko
St. Petersburg Institute for Informatics and Automation, 39, 14th Liniya, Russia,
{gor@mail.iias.spb.su, ivkote@iias.spb.su}

Abstract

The paper presents experience in application of multi-agent technology for design and implementation of multi-agent systems (MASs) intended to cooperatively solve the currently critical tasks in the area of computer network security assurance. These MASs are Agent-based Simulator of Attacks against Computer Networks, Multi-agent Intrusion Detection System and Multi-agent Intrusion Detection Learning System. Each of these MASs is based on strict formal frameworks proposed by authors and designed and implemented as software prototypes on the basis of common technology and software tool "Multi-agent System Development Kit" developed by authors. The paper sketches the above MASs and analyses advantages of use of multi-agent architecture for computer network assurance.

1. Introduction

During several last years the computer network security is a problem of big concern within information technology research area. Increasing of networks scale and intensive emerging of new information technologies, and other factors enhance the number of possible targets for attacks against computer network. All above factors negatively influence upon the efficiency of the existing network protection systems and enable research and development of *new protection models and technologies.*

Along with conventionally used security tools like firewalls, *intrusion detection systems* (IDSs) are becoming of great significance. It is well known that modern real-time IDSs are not able to detect sophisticated attacks carried out by professionals. On the other hand, IDS often interprets normal operation of networks as hostile actions producing many false alarms.

A remarkable increase of IDS efficiency could be achieved in case of using knowledge resulting from generalization and formalization of the accumulated experience regarding computer system vulnerabilities and attack cases. This is a cogent argument for the necessity of deep study and research of the essence and peculiarities of distributed attacks. The study cannot be only restricted by generalization of the experience; it has also to be based on using of formal models of attacks and *attack simulation tools*. These models and tools could be very valuable in the design of IDS capable to operate with high-level notions like "identification of an attack scenario", "forecasting of the attack development", etc. Such capabilities could make feasible to break on-line an attack development before the irreversible consequences. Besides, attack simulation tools could play an important role in validation of security policies.

The current state-of-the art in the area IDSs forces researchers and developers to focus on the elaboration of such systems that "would be capable to learn detection of new attacks and counter-measures in a semi-automatic mode in order to eliminate, as much as possible, the manual and ad-hoc elements from the process of building an intrusion detection system" [15]. Although *intrusion detection learning* is the problem that is researched about last seven years, nevertheless its importance is permanently growing. In order to detect attacks against a particular host or against the computer network as a whole, it is necessary to solve a large-scale data fusion task [2].

Contemporary view on the problem of information security is concerned with an idea that particular protective mechanisms and corresponding software must be integrated into a distributed system of autonomous software entities interacting via message exchange and making decisions in a cooperative and coordinated manner. These software entities should be adaptive to network traffic variations, to reconfiguration of the network software and hardware components and to unknown types of attack [8].

* This research is being supported by grants 01-01-108 of Russian Foundation of Basic Research and European Office of Aerospace R&D (Projects #1994 P)

The paper is focused on the formal frameworks, architectures and implementations of case studies aiming at the exploration of *potential advantages of multi-agent architecture* in network security assurance. We built three case studies: Agent-based Simulator of Attacks against Computer Networks (ASACN); Multi-agent Intrusion Detection System (MIDS); and Multi-agent Intrusion Detection Learning System (MIDLS). The chosen strategy of the network security applications implementation was based on the development of specialized software tool that could provide reusability of the most part of software for design a wide range of agent-based network security systems. It resulted in the development of the so-called "Multi-agent System Development Kit" (MASDK).

The rest of the paper is structured as follows. *Section 2* outlines the basic phases of the supported MASs design technology, generic architecture of agents generated by MASDK and a scheme of interaction of multi-agent systems developed. *Section 3* describes the developed attack simulation model and the high-level architecture of the ASACN case study. *Section 4* gives an outline of the MIDS case study. *Section 5* describes the architecture of MIDLS and software implementation of its basic components. *Section 6* presents a short overview of the existing research relevant to the paper. *Section 7* outlines the results and suggests directions of future efforts.

2. Design technology, agent architecture and scheme of the developed MASs cooperation

According to the developed technology the design and implementation of MASs for network security assurance suppose to solve two high level tasks [9]: development of the *System Kernel* of the MAS, and cloning of the software agents comprising MAS and detachment of the generated MAS from System Kernel. To specify System Kernel, two components of the developed software tool are used: (1) *Generic Agent* that aims to support modelling high-level specification of agent classes called at this stage as Generic agent subclasses; (2) *Multi-agent System Development Kit* (MAS DK) that is used in modelling of the application-oriented architecture, data, knowledge, and communication component [9].

The MAS agents generated by MAS DK have the same architecture. Differences are reflected in content of particular agents data and knowledge bases. Each agent interacts with other agents, environment which is perceived, and, possibly, modified by agents, and user communicating with agents through his interface. *Receiver of input* and *Sender of output messages* perform the respective functions. Messages received are recorded in *Input message buffer*. The order of its processing is managed by *Input message processor*. In addition, this component performs syntax analysis and KQML messages interpretation and extracts the message contents.

The component *Database of agent's dialogs* stores for each input message its attributes like identifiers, type of message and its source. If a message supposes to be replied it is mapped the respective output message when it is sent. *Meta-state machine* manages the semantic processing of input messages directing it for processing by the respective *State machines.* Another functionality of this component is allocation of management of the parallel performance of agent's processes. The basic computations of agent corresponding its role in MAS are executed by a set of *State machines* implemented as automata realizing a scenario of processing of input messages. Each agent class is provided with a set of particular message templates according to its functionalities.

The developer carries out the specialization procedure with *Editor of message templates*, which, in turn, is a component of MAS DK. The message templates are specified in KQML language and specialization corresponds to the assignment to each template the respective performatives. Communication component of each agent includes also data regarding potential addressees of messages for given template. The last data are assigned at the phase of agent class instances cloning. Message content is specified in XML.

The software code is written on the basis of Visual C++, JAVA 2 and XML software development kits. The design and implementation of all three multi-agent case studies for network security assurance is being carried out on the basis of MASDK.

The common scheme of interaction of MASs for network security assurance is depicted in figure 1. ASACN simulates the input traffic, i.e. a mixture of normal and abnormal streams of events. The abnormal stream of events is simulating attacks against the computer network. The input traffic can correspond to a reasonable sequence of these "single-phase" attacks using different

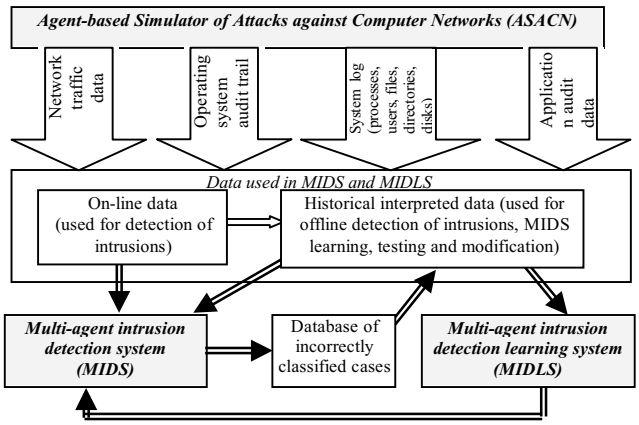

Figure 1. Interaction of the input data, intrusion detection and intrusion detection learning system

entry points (hosts). MIDS is responsible for detection of attacks against the computer network. MIDLS is for multi-level learning based on the interpreted data from the same sources and represented in the same structures as the ones used by the MIDS. In the case studies we used different configurations of the network. This network can comprise several segments of a LAN and the input traffic can be both inside and outside LAN traffic.

3. Agent-based simulator of attacks against computer networks

In the developed ASACN, distributed attack is specified as a sequence of coordinated actions of the distributed malefactors. Each malefactor is mapped as an intelligent agent of the same architecture possessing the similar functionality. While performing a distributed attack, malefactors interact to coordinate their activity.

When implementing the complex coordinated attacks, the special meta-agent forms the common scenario of attack and assigns areas of responsibility to other agents based on the general attack goal installed by user. The agents, responsible for separate fragments (steps) of the common scenario, can in turn "employ" other agents or realize separate operations independently. For this purpose the special scenarios of operations and protocols of messaging are used. The concrete scenario and protocol is determined with usage of the network attacks ontology depending on a type of the realizable goal (intention) and the attacked network response. All set of the involved agents realizing the concrete scenario compose a hierarchical structure (figure 2).

The ASACN agent (malefactor) performs an attack according to a *scenario* represented in high-layer terms. Every phase of the scenario can be detailed in terms of sub-goals of the lower layer. At the lowest one malefactor tries to achieve every upper layer sub-goal through a

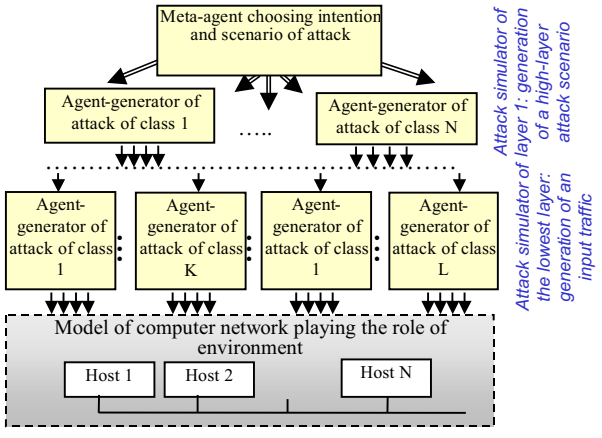

Figure 2. A hierarchical structure of ASACN at implementation of the complex attacks

sequence of acts (commands).

In the developed model we use *the malefactor's intention-centric approach to the specification* of its activity. This means that basic notions of the domain correspond to the malefactor intentions and all other notions are structured accordingly to the structure of intentions.

Attack task determines the class of scenarios that lead to the intended result. Attack task specification is as follows: *<Network (host) address, Malefactor's intention, Information about network (host), Attack object>*. The basic *malefactor's intentions* are "Resource Enumeration", "Gaining Access to Resources", "Escalating Privilege", etc. *Attack object* corresponds to the optional variable in attack task specification.

Attack formal model is represented as knowledge base, which is shared by all agents and structured according to the developed domain ontology. The developed ontology includes detailed description of the "Network attack" domain in which the notions of the bottom layer ("terminals") can be specified in terms of audit data and network packets.

Mathematical model of network attacks is specified in terms of the set of *stochastic formal grammars* interconnected through "*substitution*" operations: $M_A = <\{ G_i \}, \{ Su \}>$, where *{G}* – the formal grammars, *{Su}* – the "substitution" operations. The sequences of symbols generated by *each* of such grammars correspond to the sequences of time ordered malefactor's intentions and/or actions represented at a layer of details.

Every formal grammar is specified by quintuple $G=<V_N, V_T, S, P, A>$, where G is the grammar identifier (name), V_N is the set of non-terminal symbols (that are associated with the upper and the intermediate levels of representation of the steps of an attack scenario), V_T is the set of its terminal symbols (that designate the steps of a lower-level attack), $S \in V_N$ is the grammar axiom (an initial symbol of an attack scenario), P is the set of productions (production rules) that specify the refinement operations for the attack scenario through the substitution of the symbols of an upper level node by the symbols of the lower-level nodes, and A is the set of attributes and algorithms of their computation. The grammar production is recorded as follows: *(U)* $X \rightarrow aY$ *(Prob)*, where U is the condition for upholding the rule, X, Y is a non-terminal symbol, a is a terminal symbol, *Prob* is the probability of the rule being chosen.

Attribute component of each grammar serves for several purposes. The first of them is to specify *randomised choice of a production* at the current inference step if several productions have the equal left part non-terminals coinciding with the active non-terminal in the current sequence under inference. Also the attribute component is used to check *conditions determining the*

admissibility of using a production at the current step of inference.

The family of state machines implements algorithmic interpretation of the attack generation specified as a family of formal grammars. The peculiarity of any attack is that the malefactor's strategy depends on the results of the intermediate actions. This is the reason why the malefactor's action has to be generated on-line in parallel with the getting reaction of the attacked network. The proposed stochastic grammar syntax provides the model with this capability.

4. Muti-agent intrusion detection system

MIDS makes decisions based on the multi-level input data processing using the meta-classification scheme. The developed case study is an implementation of a particular simplified case of this architecture. The host-based part of the implemented MIDS architecture comprises the following basic components (figure 3).

Agent-demon AD-E is responsible for the input traffic pre-processing. It monitors traffic and extracts sequences of so-called "events" that are semantically meaningful from intrusion detection viewpoint. These events are sorted, stored in *AD-E* database and forwarded to the posterior processing to one or several agents.

Agent-demons for identification and authentication (*AIA*) and *for access control* (*ACA*) perform their conventional functionalities, record the results into their data bases and send messages to the *Intrusion detection agent* (IDA) if a suspicious behaviour or attempt of an attack has been detected.

Agent-demons AD-P1 and *AD-P2* are responsible for extraction of the "meaningful" patterns of "events" and making decisions regarding to the user's behaviour. In the implemented case study *AD-P1* agent is responsible for extracting patterns associated with the attacks like *finger search* and *buffer overflow*. The agent *AD-P2* is intended to extract patterns corresponding to the *denial of service* attack and *port scanning*.

*IDA*s make rule-based decisions on the basis of input facts contained in the received messages. They can receive messages about detected suspicious behaviour from agents of the same host as well as other hosts. *IDA1* processes situations like *combined spoofing attacks*, whereas *IDA2* performs high-level data processing in order to detect multi-phase attacks.

In brief, three levels of processing to detect an attack can be differentiated (see figure 3): (1) Pre-processing of input traffic by *AD-E* aimed at transformation of the input stream into sequences of significant events; (2) Extracting predefined patterns and detecting the single-phase ("simple") attacks by *AIA, ACA, AD-P1* and *AD-P1*; (3) Detecting combined attacks by *IDA1* and *IDA2*.

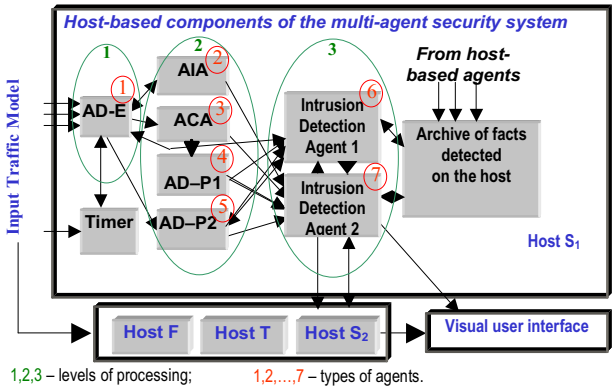

Figure 3. Architecture and interaction of host-based part of MIDS

The results of testing of the MIDS prototype exhibited the capability of the system to detect multi-phases distributed attacks performed from different source computers (*IP*-addresses) and through several entry points of the computer network.

5. Multi-agent intrusion detection learning system

The main peculiarities and resulting problems of intrusion detection learning technology result from the peculiarities of learning data. The most significant of them result from distributed nature and heterogeneity of data for intrusion detection learning. The data can be represented in different data structures and measured in different measurement scales, be of different accuracy and reliability, they may be incomplete and uncertain and, contain missing values, etc.

MIDLS includes several copies of the following classes of agents: learning data management agents; classifier testing agents; meta-data forming agents; and learning agents (figure 4).

The *learning data management agents* are intended for allocation the training and testing data between different copies of learning agents depending on their roles in the general decision-making structure. In that, *an agent's role* is *a class of attacks* it is designed to detect and also its *place* in the general decision-making *structure*. At the lower levels, base classifiers make the decisions. There may be several of them for the same subset of attacks, but they should learn on the basis of different sets of training and testing data. At the upper levels, decisions made by the base classifiers are used for making the final decision through combining the decisions made by the base classifiers. This is done by the meta-classifiers. A data management agent possesses information regarding the composition and structure of the entire amount of training and testing data, the composition and the roles of agents in

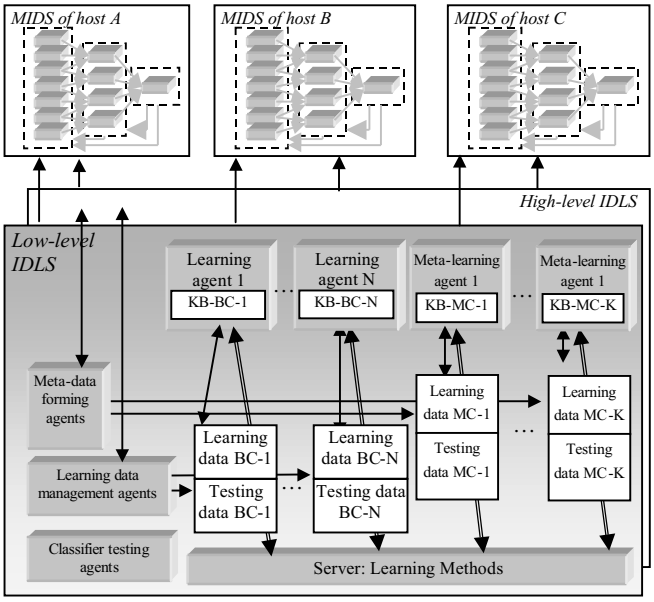

Figure 4. MIDLS architecture

the MIDS within a host. Based on this information and on its own knowledge, the agent has to propose a decision regarding the partition of the learning sample among different agents of the host.

The responsibility of the *classifier testing agents* consists in testing of classifiers based on the data sample chosen as testing and in assessing the learning quality of a classifier based on a specified set of criteria. The second task of these agents consists in launching the "additional learning" procedures if the learning results do not meet the specified criteria.

The *meta-data forming agents* possess the knowledge concerning to the meta-classifiers for which each of them has to form meta-data for training and testing. This knowledge concerns to the subset of base classifers, which decisions to be combined. Also each meta-data forming agent has information concerning to the data that serves as the basis for forming the required meta-data sample. This agent's main function consists in launching the testing the required base classifiers and recording the test results in the required format into the database used for meta-data storage.

The *learning agents* realize the main functions of the MIDLS. Two classes of the learning agents are used. The *first class of the learning agents* is designed for the task in which training and testing data are represented as ordered temporal sequences of random length. *The second class of learning agents* is designed for the learning classifiers that work with the training and testing data represented in the form of attribute vectors. This is the first distinction between the learning agent classes. The second distinction between them lies in the structures of knowledge that they extract from the learning data. Agents of the first class are designed for extracting patterns and frequent episodes. The main functions of the second class of agents consist in extracting rules from the data with attributes represented as vector. The chosen multitude of learning methods includes both the widely known methods (e.g., ID3, C4.5, boosting, and the meta-classification methodology) and the methods that have been or are being developed by the authors (e.g., visual classification method, the INFORM algorithm, Algebraic Bayes Networks).

6. Related works

Let us consider the related works in the following order: (1) attack simulation tools; (2) IDS; (3) intrusion detection learning systems.

Till now a lot of data about different security incidents is accumulated. There are a number of publications in which attack cases are systematized as taxonomies (for example, [4], [11], [14], etc.). Nevertheless, there are no serious attempts to generalize the accumulated data in order to develop a *formal model and simulator of attacks against computer network*. The publications reflect a beginning phase of research (see, for example, [3], [6], [12]). We developed a strict formal model and techniques for attack modelling based on stochastic formal-grammar-based specification of the scenarios of network attacks on the macro and micro levels.

Recently a number of new models and innovations for IDSs were proposed. Common ideas are: (1) use of knowledge-based frameworks, for example, rule-based systems, neural networks, genetic algorithms, human-like immunology systems; and (2) use of cooperation of the particular distributed components of the network security system that should allow to detect unknown distributed attacks against computer network on a whole. So multi-agent model in IDS design attracted a great attention during several last years ([1], [10], [13], etc.). It was ascertained that such a model allows enhancing IDS performance and reliability as compared with conventional approaches. However, the majority of known research using the multi-agent model of IDS only considers the simplified version of agents and their cooperative behaviour. In particular, the proposed models and architectures use agents at the pre-processing phase of the protection task. Here the agents are not knowledge-based, they are managed by a high-level software manager and do not cooperate with other security system components. We implemented the IDS possessing main advantages of intelligent multi-agent systems.

Learning intrusion detection is the problem that is researched about last seven years. The basic peculiarity of the approach developing by group of S.Stolfo ([16], [17]) is that in it intrusion detection learning is considered as a *data analysis process*. An alternative approach to the design and development of the modern learnable intrusion

detection systems is being developing in the framework of so-called "*computer immunology*" ([5], [7], etc.). This approach seems to be very perspective one but now it is at the stage of fundamental research and is exploring usefulness of implementation of some simple basic principles within computer security task. *The approach that is being elaborated by the authors* is in some respects close to the approach developed by the group of S.Stolfo. However, this similarity concerns only with the ideas in general. The distinctions are manifold. The *first* of them is in architecture of agents and MAS as a whole. The particular agents are of different specialization. The *second* distinction is that agents are supposed to interact on the basis of shared knowledge represented in ontology, which includes structured problem and subject domain components. The latter makes it possible to deal with detection not only simple attacks, but also with detection of distributed attacks against computer network as a whole. The *third* distinction is in techniques for mining knowledge from audit data that we implemented. Together with the standard techniques for learning base classifiers and meta-classifiers we implemented the original ideas and methods. Possibly, the *most important distinction* is that we consider network intrusion detection task as multi-sensor data fusion.

7. Conclusion

The paper presents the developed multi-agent network security assurance applications developed by software tool called MAS DK that aims at supporting for the basic phases of MAS technology. *ASACN* makes it possible to specify and to simulate distributed attacks at various layers of details using a strict formal model of attack scenario. The variance of attacks is ensured by the random choice of the grammar productions (or the state machine transition rules). The most significant *MIDS* advantage is a capability of comparatively "light" components of a multi-agent security system to cooperate. At present the only way to detect efficiently a distributed attack against a computer network is a cooperation of security agents distributed over the hosts of the network. *MIDLS* is viewed as a multi-sensor and a multi-level data fusion system. This system makes decisions on the basis of a multi-level model of network traffic and host-based audit data. The future research is intended to expand the capabilities of multi-agent network security assurance applications and MAS DK.

8. References

[1] J.S.Balasubramaniyan, J.O.Garcia-Fernandez, D.Isacoff, E.Spafford, and D.Zamboni, *An Architecture for Intrusion Detection Using Autonomous Agents.* Coast TR 98-05. West Lafayette, COAST Laboratory, Purdue University, 1998.

[2] T.Bass, "Intrusion Detection System and Multisensor Data Fusion: Creating Cyberspace Situational Awareness", *Communication of the ACM*, Vol.43, No. 4, 2000.

[3] S.-D.Chi, J.S.Park, K.-C.Jung and J.-S.Lee, "Network Security Modeling and Cyber Attack Simulation Methodology", *Lecture Notes in Computer Science*, Vol.2119, 2001.

[4] F.B.Cohen, "Information System Attacks: A Preliminary Classification Scheme", *Computers and Security*, Vol.16, No.1, 1997.

[5] D.Dasgupta and F.Gonzales, "An Intelligent Intrusion Detection System for Intrusion Detection", *Lecture Notes in Computer Science,* Vol.2052, Springer Verlag, 2001.

[6] S.T.Eckmann, G.Vigna, and R.A.Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection", *Proceedings of the ACM Workshop on Intrusion Detection*, Athens, Greece, November 2000.

[7] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology", *Communications of the ACM,* Vol. 40, No. 10, 1997.

[8] V.Gorodetski, I.Kotenko, L.Popyack, and V.Skormin, "Integrated Multi-Agent Information Security System: Mechanisms of Agents' Operation and Learning", *PAAM' 2000.* Manchester. UK. 2000.

[9] V.Gorodetski, O.Karsayev, I.Kotenko, A.Khabalov, "Software Development Kit for Multi-agent Systems Design and Implementation", *Lecture Notes in Artificial Intelligence,* Vol.2296, Springer Verlag, 2002.

[10] G.Helmer, J.Wong, V.Honavar, and L.Miller, Intelligent Agents for Intrusion Detection, *Proceedings of the 1998 IEEE Information Technology Conference, Environment for the Future.* Syracuse. NY: IEEE, 1998.

[11] J.D.Howard, and T.A.Longstaff, *A Common Language for Computer Security Incidents*, SANDIA REPORT, SAND98-8667, October 1998.

[12] M.-Y.Huang, and T.M.Wicks, "A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis". *Proceedings of Second International Workshop on Recent Advances in Intrusion Detection, RAID'98*, Louvain-la-Neuve, 1998.

[13] W.Jansen, P.Mell, T. Karygiannis, and D.Marks, "Mobile Agents in Intrusion Detection and Response", *Proceedings of the 12th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, June 2000.

[14] I.V.Krsul, *Software Vulnerability Analysis*, Ph.D. Dissertation, Computer Sciences Department, Purdue University, Lafayette, IN, May, 1998.

[15] W.Lee, S. J.Stolfo, and K.Mok, "Mining audit data to build intrusion detection models", *Proceedings of 4th International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 1998.

[16] W.Lee, S.Stolfo, and K.Mok, "Algorithms for Mining System Audit Data". *Data Retrieval and Data Mining.* T. Y. Lin and N. Cercone (eds.), Kluwer Academic Publishers, 1999.

[17] A.Prodromidis, P.Chan, and S.Stolfo, "Meta-Learning in Distributed Data Mining Systems: Issues and Approaches", *Advances in Distributed Data Mining*, Kargupta and Chan (eds.), AAAI Press, 1999.