Designing Intrusion Detection Systems: Architectures, Challenges and Perspectives

Srinivas Mukkamala, Andrew Sung and Ajith Abraham*

Department of Computer Science, New Mexico Tech, USA
*Department of Computer Science, Oklahoma State University, USA

Abstract:
Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability. 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. Due to increasing incidents of cyber attacks, building effective intrusion detection systems are essential for protecting information systems security, and yet it remains an elusive goal and a great challenge. We present the state-of-the-art of the evolution of intrusion detection systems and address some of the research challenges to design efficient and effective intrusion detection systems. Further distributed intrusion detection systems are presented which could be used to detect and prevent attacks that would be invisible to any single system or whose significance would be missed if information from only a single system were available. We finally illustrate how a data mining approach could reduce abundant/redundant and noisy data and design effective intrusion detection systems.

1 Introduction

Intrusion detection is a problem of great significance to critical infrastructure protection owing to the fact that computer networks are at the core of the nation's operational control. This paper summarizes the use of different artificial intelligent techniques to build efficient IDSs in terms of classification accuracy, learning time and testing time. Since the ability of a good detection technique gives more accurate results, it is critical for intrusion detection in order for the IDS to achieve maximal performance. Therefore, we study different intelligent computing techniques for building models based on DARPA intrusion detection data illustrated in Figure 1[1]. This paper summarizes the performance of Intrusion Detection Systems (IDSs) built using Artificial Neural Networks or ANNs [2], Support Vector Machines or SVMs [3], Multivariate Adaptive Regression Splines or MARS [4] and Liner Genetic Programs (LGPs) [5]. Single point or centralized IDSs may have packet loss on heavy load conditions on Gigabit networks, such a situation will hinder the performance of an IDS. Distributed computational intelligent agents distribute the processing effort of detecting intrusions among multiple agents that are distributed on the network; that help detect intrusions that are missed by the centralized IDS.

Since most of the intrusions can be uncovered by examining patterns of user activities, many IDSs have been built by utilizing the recognized attack and misuse patterns that can classify a user's activity as normal or abnormal (attack). Data mining techniques are used to analyze audit trails and extract knowledge (features) that help in distinguishing intrusive patterns form normal patterns. Feature rankings algorithms help in deciding which features are important for classifying intrusions and which ones are redundant or noisy. Several intelligent techniques including but not limited to ANNs, SVMs, Petri nets, and data mining techniques are being used to build intrusion detection systems. In this paper, we will concentrate on using SVMs,

MARS, LGPs and ANNs with different training functions to achieve better classification accuracies. The data we used in our experiments originated from Lincoln Lab. It was developed for intrusion detection system evaluations by DARPA and is considered a benchmark for intrusion detection evaluations [1].

In the rest of the paper, a brief introduction to related work in the field of intrusion detection is given in section 2. A brief introduction to computer attack taxonomy and the data we used is given in section 3. In section 4 we describe the theoretical aspects of data mining, computational intelligence, ANNs, SVMs, and MARS and LGPs. Implementation of distributed computational intelligent agents for intrusion detection is also described in section 4. In section 5 we present the experimental results of ANNs, SVMs, MARS and distributed computational intelligent agents to detect probes. In section 6 we describe the importance of feature ranking for IDS and summarize the results obtained using the important features for classifying intrusions. In section 7, we summarize our experience of using artificial intelligent techniques to build IDSs.

2 Intrusion Detection Systems and Related Research

Identifying unauthorized use, misuse and attacks on information systems is defined as intrusion detection [6,7]. The most popular way to detect intrusions has been done by using audit data generated by operating systems and by networks. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. It is important to analyze the audit data even after an attack has occurred, for determining the extent of damage occurred, this analysis helps in attack trace back and also helps in recording the attack patterns for future prevention of such attacks. An intrusion detection system can be used to analyze audit data for such insights. This makes intrusion detection system a valuable real-time detection and prevention tool as well as a forensic analysis tool.

Soft computing techniques are being widely used by the IDS community due to their generalization capabilities that help in detecting know intrusions and unknown intrusions or the attacks that have no previously described patterns. Earlier studies have utilized a rule based approach for intrusion detection, but had a difficulty in identifying new attacks or attacks that had no previously describe patterns [8-11]. Lately the emphasis is being shifted to learning by examples and data mining paradigms. Neural networks have been extensively used to identify both misuse and anomalous patterns [12-14]. Recently support vector machines and their variants are being proposed to detect intrusions [14]. Several researchers proposed data mining techniques to identify key patterns that help in detecting intrusions [15-17]. Distributed agent technology is being proposed by a few researchers to overcome the inherent limitations of the client-server paradigm and to detect intrusions in real time [18-21].

2.1 Misuse Detection

The idea of misuse detection is to represent attacks in the form of a pattern or a signature so that the same attack can be detected and prevented in future. These systems can detect many or all known attack patterns [22], but they are of little use for detecting naive attack methods. The main issues of misuse detection is how to build signatures that include possible signatures of attacks build a signature that includes all possible variations of the pertinent attack to avoid false negatives, and how to build signatures that do not match non-intrusive activities to avoid false positives.

2.2 Anomaly Detection

The idea here is that if we can establish a normal activity profile for a system, in theory we can flag all system states varying from the established profile as intrusion attempts. However, if the set of intrusive activities is not identical to the set of anomalous activities, the situation becomes more interesting instead of being exactly the same, we find few interesting possibilities. Anomalous activities that are not intrusive are flagged as intrusive, though they are false positives. Actual intrusive activities that go undetected are called false negatives. This is a serious issue, and is far more serious than the problem of false positives. One of the main issues of anomaly detection systems is the selection of threshold levels so that neither of the above problems is unreasonably magnified. Anomaly detection is usually computationally expensive because of the overhead of keeping track of and possibly updating several system profiles. Recent proposed system LEARD (Learning Rules for Anomaly Detection) discovers relationships among attributes in order to model application protocols [23,24].

3. Computer Attack Taxonomy

A good attack taxonomy makes it possible to classify individual attacks into groups that share common properties. Taxonomy should hold the property; if classifying in one category excludes all others because categories do not overlap. The taxonomy of the data we used in our experiments consists of four groups probing, denial of service, gaining higher level privileges and accessing a local machine from a remote site with out proper privileges.

3.1 Probing

Probing is a class of attacks 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. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise. Different types of probe attacks are illustrated in Table 1.

Table 1 Probe Attacks

Attack Type	Service	Mechanism	Effect of the attack
Ipsweep	Icmp	Abuse of feature	Identifies active machines
Mscan	Many	Abuse of feature	Looks for known vulnerabilities
Nmap	Many	Abuse of feature	Identifies active ports on a machine
Saint	Many	Abuse of feature	Looks for known vulnerabilities
Satan	Many	Abuse of feature	Looks for known Vulnerabilities

3.2 Denial of Service Attacks

Denial of Service (DoS) is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch DoS attacks: by abusing the

computers legitimate features; by targeting the implementations bugs; or by exploiting the system's misconfigurations. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users. Some of the popular attack types are illustrated in Table 2.

Table 2 Denial of Service Attacks

Attack Type	Service	Mechanism	Effect of the attack
Apache2	http	Abuse	Crashes httpd
Back	http	Abuse/Bug	Slows down server response
Land	http	Bug	Freezes the machine
Mail bomb	N/A	Abuse	Annoyance
SYN Flood	TCP	Abuse	Denies service on one or more ports
Ping of Death	Icmp	Bug	None
Process table	TCP	Abuse	Denies new processes
Smurf	Icmp	Abuse	Slows down the network
Syslogd	Syslog	Bug	Kills the Syslogd
Teardrop	N/A	Bug	Reboots the machine
Udpstrom	Echo/ Chargen	Abuse	Slows down the network

3.3 User to Root Attacks

User to root 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. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions. Please refer to Table 3 for some of the attack types in this category.

Table 3 User to Super-User Attacks

Attack Type	Service	Mechanism	Effect of the attack
Eject	User session	Buffer overflow	Gains root shell
Ffbconfig	User session	Buffer overflow	Gains root shell
Fdformat	User session	Buffer overflow	Gains root shell
Loadmodule	User session	Poor environment sanitation	Gains root shell
Perl	User session	Poor environment sanitation	Gains root shell
Ps	User session	Poor Temp file management	Gains root shell
Xterm	User session	Buffer overflow	Gains root shell

3.4 Remote to User Attacks

A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2U attacks; the most common attack in this class is done using social engineering. Some of the R2U attacks are presented in Table 4.

Table 4 Remote to User Attacks

Attack Type	Service	Mechanism	Effect of the attack
Dictionary	telnet, rlogin, pop, ftp, imap	Abuse feature	Gains user access
Ftp-write	ftp	Misconfig.	Gains user access
Guest	telnet, rlogin	Misconfig.	Gains user access
Imap	imap	Bug	Gains root access
Named	dns	Bug	Gains root access
Phf	http	Bug	Executes commands as http user
Sendmail	smtp	Bug	Executes commands as root
Xlock	smtp	Misconfig.	Spoof user to obtain password
Xnsoop	smtp	Misconfig.	Monitor key stokes remotely

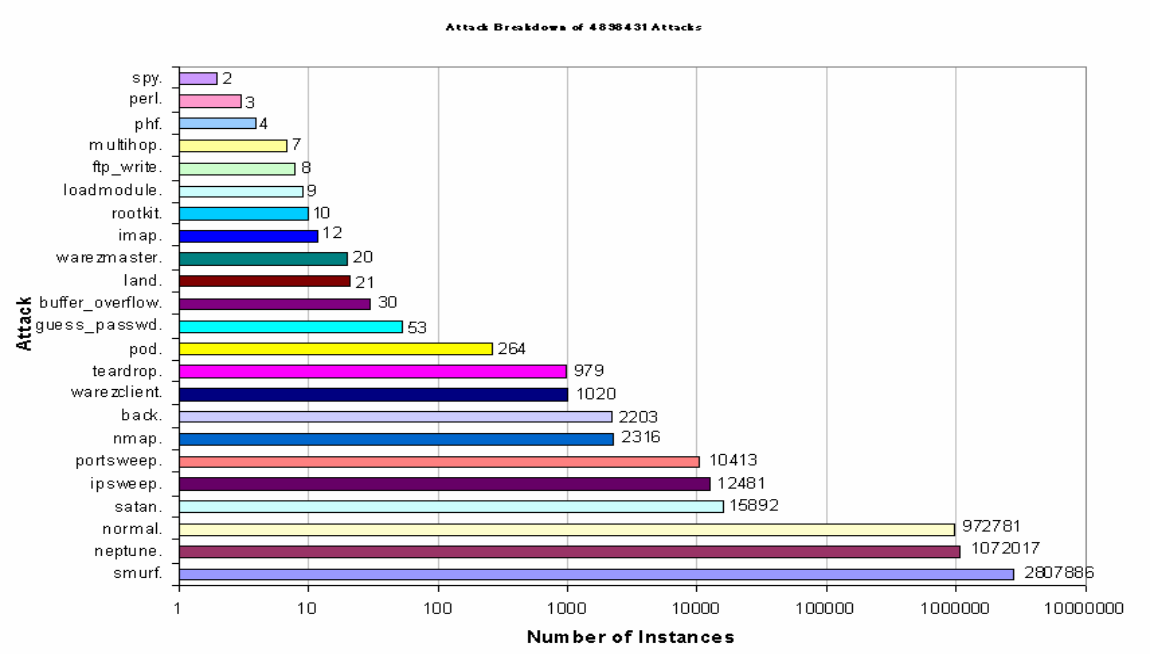

Figure 1: Intrusion Detection Data Distribution

4 Data Mining and Computational Intelligence

Data mining (also known as Knowledge Discovery in Databases - KDD) has been defined by Frawley etl [25] as "The nontrivial extraction of implicit, previously unknown, and potentially useful information from data". Data ming techniques use machine learning, statistical and visulaization techniques to discover and present knowledge from the raw information in a easily comprehensible form to humans. In the field of intrusion detection data mining programs are used to analyze audit trails and provide knowledge (features) that help in distinguishing intrusive patterns form normal activity [17,26].

Soft computing was first proposed by Zadeh [27] to construct new generation computationally intelligent hybrid systems consisting of neural networks, fuzzy inference system, approximate reasoning and derivative free optimization techniques. It is well known that the intelligent systems, which can provide human like expertise such as domain knowledge, uncertain reasoning, and adaptation to a noisy and time varying environment, are important in tackling practical computing problems. In contrast with conventional Artificial Intelligence (AI) techniques which only deal with precision, certainty and rigor the guiding principle of hybrid systems is to exploit the tolerance for imprecision, uncertainty, low solution cost, robustness, partial truth to achieve tractability, and better rapport with reality.

4.1 Artificial Neural Networks (ANNs)

The artificial neural network (ANN) methodology enables us to design useful nonlinear systems accepting large numbers of inputs, with the design based solely on instances of input-output relationships. Artificial neural network (in the present context, multilayer, feed forward type networks) consists of a collection of highly-interconnected processing elements to perform an input-output transformation. The actual transformation is determined by the set of weights associated with the links connecting elements. The neural network gains knowledge about the transformation to be performed by iteratively learning from a sufficient training set of samples or input-output training pairs. A well-trained network can perform the transformation correctly and also possess some generalization capability.

Since multi-layer feed forward ANNs are capable of making multi-class classifications, an ANN is employed to perform the intrusion detection, using the same training and testing sets as those for other connectionist paradigms.

4.1.1 Resilient Back propagation (RP)

The purpose of the resilient back propagation training algorithm is to eliminate the harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased by a factor whenever the derivative with respect that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating the weight change will be reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased [28].

Let us report only one learning algorithm and we need to write a bit in detail. I can help you a bit here.

4.2 Support Vector Machines (SVMs)

The SVM approach transforms data into a feature space *F* that usually has a huge dimension. It is interesting to note that SVM generalization depends on the geometrical characteristics of the training data, not on the dimensions of the input space [29]. Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Vapnik shows how training a SVM for the pattern recognition problem leads to the following quadratic optimization problem [30].

$$\text{Minimize: } W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \qquad (4)$$

$$\text{Subject to } \sum_{i=1}^{l} y_i \alpha_i \qquad (5)$$
$$\forall i : 0 \le \alpha_i \le C$$

Where l is the number of training examples α is a vector of l variables and each component α_i corresponds to a training example (x_i, y_i). The solution of (4) is the vector α^* for which (4) is minimized and (5) is fulfilled.

4.3 Multivariate Adaptive Regression Splines (MARS)

Splines can be considered as an innovative mathematical process for complicated curve drawings and function approximation. To develop a spline the X-axis is broken into a convenient number of regions. The boundary between regions is also known as a knot. With a sufficiently large number of knots virtually any shape can be well approximated. The MARS model is a regression model using basis functions as predictors in place of the original data. The basis function transform makes it possible to selectively blank out certain regions of a variable by making them zero, and allows MARS to focus on specific sub-regions of the data. It excels at finding optimal variable transformations and interactions, and the complex data structure that often hides in high-dimensional data [4].

Given the number of records in most data sets, it is infeasible to approximate the function $y=f(x)$ by summarizing y in each distinct region of x. For some variables, two regions may not be enough to track the specifics of the function. If the relationship of y to some x's is different in 3 or 4 regions, for example, the number of regions requiring examination is even larger than 34 billion with only 35 variables. Given that the number of regions cannot be specified a priori, specifying too few regions in advance can have serious implications for the final model. A solution is needed that accomplishes the following two criteria:

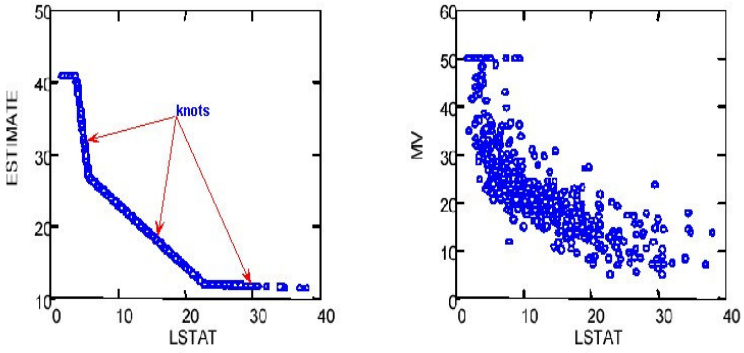

Figure 2. MARS Data Estimation Using Spines and Knots (actual data on the right)

- judicious selection of which regions to look at and their boundaries
- judicious determination of how many intervals are needed for each variable

Given these two criteria, a successful method will essentially need to be adaptive to the characteristics of the data. Such a solution will probably ignore quite a few variables (affecting variable selection) and will take into account only a few variables at a time (also reducing the number of regions). Even if the method selects 30 variables for the model, it will not look at all 30 simultaneously. Such simplification is accomplished by a decision tree at a single node, only ancestor splits are being considered; thus, at a depth of six levels in the tree, only six variables are being used to define the node.

4.3.1 MARS Smoothing, Splines, Knots Selection and Basis Functions

To estimate the most common form, the cubic spline, a uniform grid is placed on the predictors and a reasonable number of knots are selected. A cubic regression is then fit within each region. This approach, popular with physicists and engineers who want continuous second derivatives, requires many coefficients (four per region) to be estimated. Normally, two constraints, which dramatically reduce the number of free parameters, can be placed on cubic splines:

- curve segments must join,
- continuous first and second derivatives at knots (higher degree of smoothness)

Figure 2 depicts a MARS spline with three knots. A key concept underlying the spline is the knot. A knot marks the end of one region of data and the beginning of another. Thus, the knot is where the behavior of the function changes. Between knots, the model could be global (e.g., linear regression). In a classical spline, the knots are predetermined and evenly spaced, whereas in MARS, the knots are determined by a search procedure. Only as many knots as needed are included in a MARS model. If a straight line is a good fit, there will be no interior knots. In MARS, however, there is always at least one "pseudo" knot that corresponds to the smallest observed value of the predictor [31]. Finding the one best knot in a simple regression is a straightforward search problem: simply examine a large number of potential knots and choose the one with the best R^2. However, finding the best pair of knots requires far more computation, and finding the best set of knots when the actual number needed is unknown is an even more challenging task. MARS finds the location and number of needed knots in a forward/backward stepwise fashion. A model which is clearly over fit with too many knots is generated first; then, those knots that contribute least to the overall fit are removed. Thus, the forward knot selection will include many incorrect knot locations, but these erroneous knots will eventually (although this is not guaranteed), be deleted from the model in the backwards pruning step.

4.4 Linear Genetic Programs (LGPs)

Linear genetic programming is a variant of the GP technique that acts on linear genomes [15,16]. 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 ++). An alternate approach is to evolve a computer program at the machine code level, using lower level representations for the individuals. This can tremendously hasten up the evolution process as, no matter how an individual is initially represented, finally it always has to be represented as a piece of machine code, as fitness evaluation requires physical execution of the individuals. The basic unit of evolution is a native machine code instruction that runs on the floating-point processor unit (FPU). Since different instructions may have different sizes, here instructions are clubbed up together to form instruction blocks of 32 bits each. The instruction blocks hold one or more native machine code instructions, depending on the sizes of the instructions. A crossover point can occur only between instructions and is prohibited from occurring within an instruction. However the mutation operation does not have any such restriction.

4.5 Computational Intelligent Agents (CIA) Based Architecture

The CIA based architecture for detecting computer attacks consists of several modules that will be executed by the agents in a distributed manner. Communication among the agents is done utilizing the TCP/IP sockets. Agent modules running on the host computers consist of data collection agents, data analysis agents, and response agents. Agents running on the secure devices consist of the agent control modules that include agent regeneration, agent dispatch, maintaining intrusion signatures and information of the features and feature ranking algorithms that help identify the intrusions.

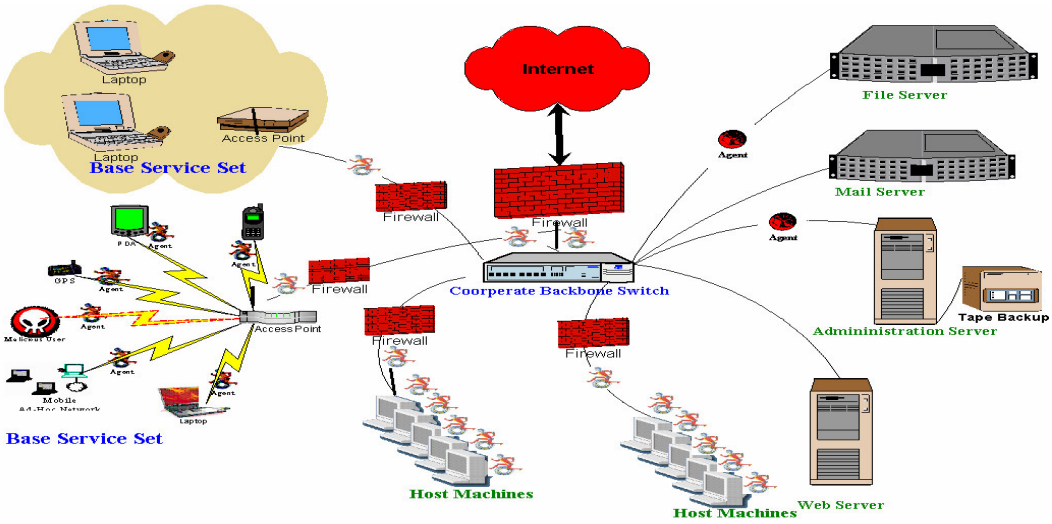

Figure 3: Computational Intelligent Agents Architecture

Host Agents: Reside on the hosts of the internal network and perform the tasks specified by the master agent. These agents are implemented to be read only and fragile. In the event of tampering or modification the agent reports to the server agent and automatically ends its life.

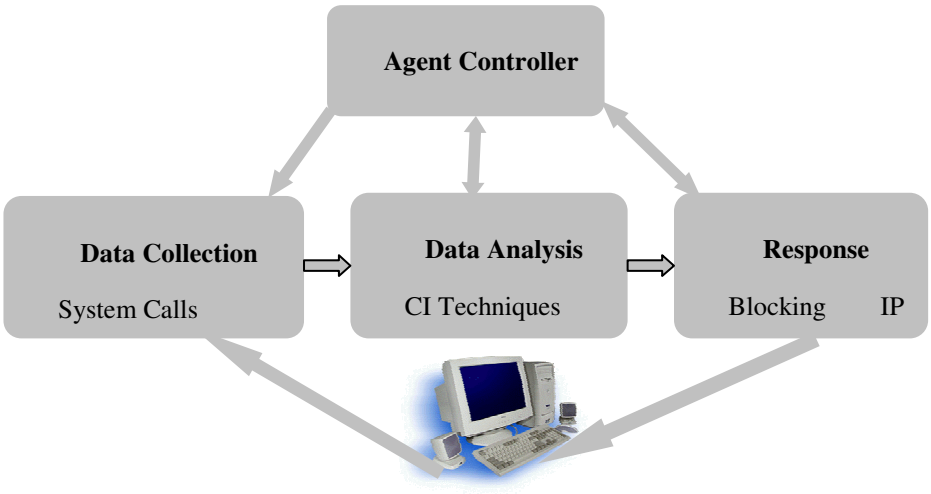

Figure 4: Functionality of Host Based Agents

Server Agents: Reside on the secure server of the network. Controls the individual host agents for monitoring the network and manages communication between the agents if necessary. These agents manage the life cycle and also update the host agents with new detection, feature extraction, response and trace mechanisms.

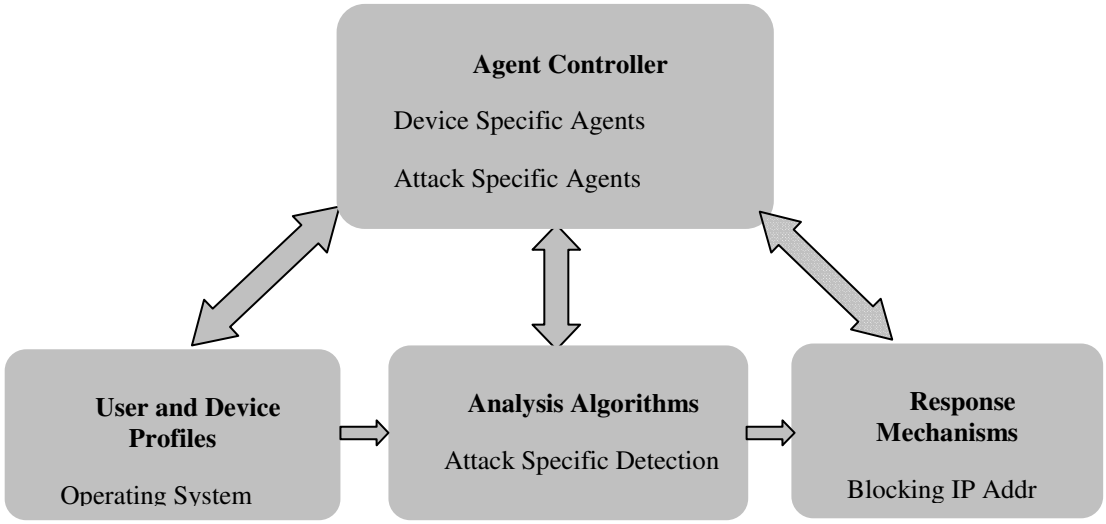

Figure 5: Functionality of Server Agents

Advantages of the proposed model:

- With prior knowledge of the device and user profiles of the network, specific agents can be designed and implemented in a distributed fashion
- Highly optimized parallel algorithms can be designed and implemented independently for data collection agents, data analysis agents, and response agents
- Analysis for computationally limited devices can be offloaded to cooperating systems and run in parallel to provide capabilities not otherwise available on such devices
- Attack-specific agents can be implemented and dispatched to respond to specific new threats
- Efficient detection algorithms can be implemented with less overhead

- Rapid intrusion response and trace back can be performed more easily with the agents communicating with each other
- Adjustable detection thresholds can be implemented

5 Experiments

(Write a bit more about the data)
In our experiments, we perform 5-class classification. The (training and testing) data set contains 11982 randomly generated points from the five classes, with the number of data from each class proportional to its size, except that the smallest class is completely included. The normal data belongs to class1, probe belongs to class 2, denial of service belongs to class 3, user to super user belongs to class 4, remote to local belongs to class 5. A different randomly selected set of 6890 points of the total data set (11982) is used for testing different intelligent techniques.

5.1 Experiments Using Neural Networks

The same data set describe in section 2 is being used for training and testing different neural network algorithms. The set of 5092 training data is divided in to five classes: normal, probe, denial of service attacks, user to super user and remote to local attacks. Where the attack is a collection of 22 different types of instances that belong to the four classes described in section 2, and the other is the normal data. In our study we used two hidden layers with 20 and 30 neurons each and the networks were trained using RP, SCG and OSS algorithms.

The network was set to train until the desired mean square error of 0.001 was met. During the training process the goal was met at 303 epochs for SCG, 66 epochs for RP and 638 epochs for OSS.

As multi-layer feed forward networks are capable of multi-class classifications, we partition the data into 5 classes (Normal, Probe, Denial of Service, and User to Root and Remote to Local). SCG performed with an accuracy of 95.25%; network using RP achieved an accuracy of 97.04%; network using OSS performed with an accuracy of 93.60%.

Table 5 Performance of the Best Neural Network Training Algorithm (RP)

	Normal	**Probe**	**DoS**	**U2Su**	**R2L**	**%**
Normal	1394	5	1	0	0	99.6
Probe	49	649	2	0	0	92.7
DoS	3	101	4096	2	0	97.5
U2Su	0	1	8	12	4	48.0
R2L	0	1	6	21	535	95.0
%	96.4	85.7	99.6	34.3	99.3	

The top-left entry of Table 6 shows that 1394 of the actual “normal” test set were detected to be normal; the last column indicates that 99.6 % of the actual “normal” data points were detected correctly. In the same way, for “Probe” 649 of the actual “attack” test set were correctly detected; the last column indicates that 92.7% of the actual “Probe” data points were detected

correctly. The bottom row shows that 96.4% of the test set said to be "normal" indeed were "normal" and 85.7% of the test set classified, as "probe" indeed belongs to Probe. The overall accuracy of the classification is 97.04 with a false positive rate of 2.76% and false negative rate of 0.20 %.

5.2 Experiments Using Support Vector Machines

The data set described in section 4 is being used to test the performance of support vector machines. Note the same training test (5092) used for training the neural networks and the same testing test (6890) used for testing the neural networks are being used to validate the performance. Because SVMs are only capable of binary classifications, we will need to employ five SVMs, for the 5-class classification problem in intrusion detection, respectively. We partition the data into the two classes of "Normal" and "Rest" (Probe, DoS, U2Su, R2L) patterns, where the Rest is the collection of four classes of attack instances in the data set. The objective is to separate normal and attack patterns. We repeat this process for all classes. Training is done using the RBF (radial bias function) kernel option; an important point of the kernel function is that it defines the feature space in which the training set examples will be classified. Table 6 summarizes the results of the experiments.

5.3 Experiments Using MARS

We use 5 basis functions and selected a setting of minimum observation between knots as 10. The MARS training mode is being set to the lowest level to gain higher accuracy rates. Five MARS models are employed to perform five class classifications (normal, probe, denial of service, user to root and remote to local). We partition the data into the two classes of "Normal" and "Rest" (Probe, DoS, U2Su, R2L) patterns, where the Rest is the collection of four classes of attack instances in the data set. The objective is to separate normal and attack patterns. We repeat this process for all classes. Table 6 summarizes the results of MARS.

5.4 Experiments Using LGPs

LGP manipulates and evolves program at the machine code level [5]. The settings of various LGP parameters are of utmost importance for successful performance of the system. This section discusses the different parameter settings used for the experiment, justification of the choices and the significances of these parameters. 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. A constant crossover rate of 90% has been used for all the simulations. Five LGP models are employed to perform five class classifications (normal, probe, denial of service, user to root and remote to local). We partition the data into the two classes of "Normal" and "Rest" (Probe, DoS, U2Su, R2L) patterns, where the Rest is the collection of four classes of attack instances in the data set. The objective is to separate normal and attack patterns. We repeat this process for all classes. Table 6 summarizes the results of the experiments using LGPs.

Table 6 Performance Comparison of Testing for 5 class Classifications

Class	% Accuracy			
	RP	**SVM**	**MARS**	**LGP**
Normal	99.57	98.42	99.71	99.71
Probe	92.71	98.57	56.42	99.86
DoS	97.47	99.45	96	99.90
U2Su	48.00	64.00	40.00	64
R2L	95.73	97.33	98.75	99.47
Overall	97.09	98.85	92.75	99.73

5.5 Experiments Using CIAs

Experiments are performed on a real network using two clients and the server that serves the computer science network. The clients had CIA installed on them to identify or detect probes that are targeted to the server we are protecting. Our primary goal in these experiments is to detect probes targeting the server we are trying to protect. Our network parser gives the summary of each connection made from a host to the server and constructs a feature set to input into a classifier for classification. The output form a classifier is either normal or probe for each connection. Nmap an open source tool is used to collect probe data. Probing is a class of attacks 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. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise. Nmap is installed on the clients that have CIA installed. A variety of probes SYN stealth, FIN stealth, ping sweep, UDP scan, null scan, xmas tree, IP scan, idle scan, ACK scan, window scan, RCP scan, and list scan with several options are targeted at the server. Normal data included multiple sessions of ftp, telnet, SSH, http, SMTP, pop3 and imap. Network data originating form a host to the server that included both normal and probes is collected for analysis; for proper labeling of data for training the classifiers normal data and probe data are collected at different times.

55.1 CIA System and Implementation

Computer probes that are intended to discover information of a computer system can be detected by careful observation of network packets. Probing tools in an effort to identify host information send connection requests to closed ports and non-existing systems. Knowledge of how a network and its hosts are being used will help in distinguishing between normal activity and probes.

The primary goal of CIA is to detect probes at the host level. Our system is integrated with three major components: a network data parser, data classifier and a response mechanism. Network data parser we developed uses the WINPCAP library to capture network packets to extract the features required for classification. Output summary of the parser includes six

features, duration of the connection to the target machine, the protocol used, the service, number of source bytes, and number of destination bytes. Feature set for our experiments is chosen based on our feature ranking results obtained using the DARPA intrusion detection evaluation data [17]. Network Parser reformats the extracted features to input a classifier to detect probes among other normal network packets. Once the feature set is being constructed it is fed into a suite of classifiers. Classifiers used in our experiments are SVMs, MARS and LGP. Output from the classifier is the classification of the connection into normal activity or probe. If a connection is classified as probe a classifier sends a message to the server using TCP/IP sockets and the boundary controllers are updated for necessary response with human intervention to block malicious activity. Table 7 summarizes the results of CIAs using different classifiers to detect stealthy probes.

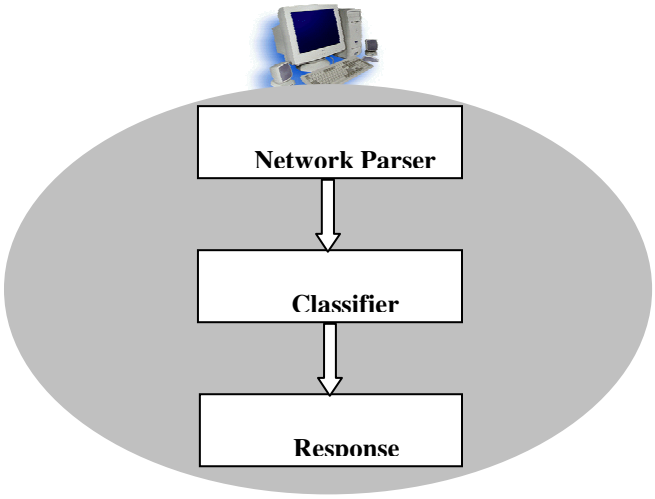

Figure 6. CIA Prototype Implementation

Table 7 Performance Comparison of Classifiers to Detect Probes Using CIAs

Class	**% Accuracy**		
	SVM	**MARS**	**LGP**
Normal	99.75	99.12	100

6 Significance of Input Features

Feature selection and ranking [17,26] is an important issue in intrusion detection. Of the large number of features that can be monitored for intrusion detection purpose, which are truly useful, which are less significant, and which may be useless? The question is relevant because the elimination of useless features (the so-called audit trail reduction) enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of an IDS. In cases where there are no useless features, by concentrating on the most important ones we may well improve the time performance of an IDS without affecting the accuracy of detection in statistically significant ways.

The feature ranking and selection problem for intrusion detection is similar in nature to various engineering problems that are characterized by:

- Having a large number of input variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of varying degrees of importance to the output **y**; i.e., some elements of **x** are essential, some are less important, some of them may not be mutually independent, and some may be useless or irrelevant (in determining the value of **y**)
- Lacking an analytical model that provides the basis for a mathematical formula that precisely describes the input-output relationship, $\mathbf{y} = \boldsymbol{F}(\mathbf{x})$
- Having available a finite set of experimental data, based on which a model (e.g. neural networks) can be built for simulation and prediction purposes

Due to the lack of an analytical model, one can only seek to determine the relative importance of the input variables through empirical methods. A complete analysis would require examination of all possibilities, e.g., taking two variables at a time to analyze their dependence or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n experiments!) and not infallible (since the available data may be of poor quality in sampling the whole input space). In the following, therefore, we apply the technique of deleting one feature at a time and support vector decision function to rank the input features and identify the most important ones for intrusion detection using SVMs. Table 8 summarizes the results of feature ranking experiments using PBRM and SVDF.

6.1 Performance-Based Ranking Method (PBRM)

We first describe a general (i.e., independent of the modeling tools being used), performance-based input ranking methodology: One input feature is deleted from the data at a time; the resultant data set is then used for the training and testing of the classifier. Then the classifier's performance is compared to that of the original classifier (based on all features) in terms of relevant performance criteria. Finally, the importance of the feature is ranked according to a set of rules based on the performance comparison.

The procedure is summarized as follows:

1. compose the training set and the testing set;

for each feature ***do*** the following

2. delete the feature from the (training and testing) data;
3. use the resultant data set to train the classifier;
4. analyze the performance of the classifier using the test set, in terms of the selected performance criteria;
5. rank the importance of the feature according to the rules;

6.1.1 Performance Metrics

To rank the importance of the 41 features (of the DARPA data) in an SVM-based IDS, we consider three main performance criteria: overall accuracy of (5-class) classification; training time; and testing time. Each feature will be ranked as "important", "secondary", or "insignificant", according to the following rules that are applied to the result of performance comparison of the original 41-feature SVM and the 40-feature SVM:

1. ***If*** *accuracy* decreases ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is important
2. ***If*** *accuracy* decreases ***and*** training time increases ***and*** testing time increases, ***then*** the feature is important
3. ***If*** *accuracy* decreases ***and*** training time decreases ***and*** testing time increases, ***then*** the feature is important

4. ***If*** *accuracy* unchanges ***and*** training time increases ***and*** testing time increases, ***then*** the feature is important
5. ***If*** *accuracy* unchanges ***and*** training time decreases *and* testing time increases, ***then*** the feature is secondary
6. ***If*** *accuracy* unchanges ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is secondary
7. ***If*** *accuracy* unchanges ***and*** training time decreases ***and*** testing time decreases, ***then*** the feature is unimportant
8. ***If*** *accuracy* increases ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is secondary
9. ***If*** *accuracy* increases ***and*** training time decreases ***and*** testing time increases, ***then*** the feature is secondary
10. ***If*** *accuracy* increases ***and*** training time decreases ***and*** testing time decreases, ***then*** the feature is unimportant

6.2 SVM-specific Feature Ranking Method

Information about the features and their contribution towards classification is hidden in the support vector decision function. Using this information one can rank their significance, i.e., in the equation

$$\mathbf{F}(\mathbf{X}) = \Sigma \mathbf{W}_i \mathbf{X}_i + \mathbf{b}$$

The point **X** belongs to the positive class if **F**(**X**) is a positive value. The point **X** belongs to the negative class if **F**(**X**) is negative. The value of **F**(**X**) depends on the contribution of each value of **X** and $\mathbf{W}_i$. The absolute value of $\mathbf{W}_i$ measures the strength of the classification. If $\mathbf{W}_i$ is a large positive value then the i^{th} feature is a key factor for positive class. If $\mathbf{W}_i$ is a large negative value then the i^{th} feature is a key factor for negative class. If $\mathbf{W}_i$ is a value close to zero on either the positive or the negative side, then the i^{th} feature does not contribute significantly to the classification. Based on this idea, a ranking can be done by considering the support vector decision function.

6.2.2 Support Vector Decision Function Ranking Method (SVDFRM)

The input ranking is done as follows: First the original data set is used for the training of the classifier. Then the classifier's decision function is used to rank the importance of the features. The procedure is:

1. Calculate the weights from the support vector decision function;
2. Rank the importance of the features by the absolute values of the weights;

Table 8 Performance of SVMs Using Important Features

Class	No of Features Identified		Training Time (sec)		Testing Time (sec)		Accuracy (%)	
	PBRM	**SVDFRM**	**PBRM**	**SVDFRM**	**PBRM**	**SVDFRM**	**PBRM**	**SVDFRM**
Normal	25	20	9.36	4.58	1.07	0.78	99.59	99.55
Probe	7	11	37.71	40.56	1.87	1.20	99.38	99.36
DOS	19	11	22.79	18.93	1.84	1.00	99.22	99.16

U2Su	8	10	2.56	1.46	0.85	0.70	99.87	99.87
R2L	6	6	8.76	6.79	0.73	0.72	99.78	99.72

7 Summary and Conclusions

Current IDS testing techniques to data are becoming increasingly complex. There are no standard testing standards that can quantify the performance of IDS in terms scalability, data handling rate, time to detect an attack, etc. Current IDS heavily rely on unencrypted audit trails and if the data is encrypted this might hinder the performance or even the IDS might be come absolute in terms of detecting intrusions. Our research has clearly shown the importance of using distributed computational intelligent agent approach for modeling intrusion detection systems.

- SVMs outperform MARS and ANNs in the important respects of scalability (SVMs can train with a larger number of patterns, while would ANNs take a long time to train or fail to converge at all when the number of patterns gets large); training time and running time (SVMs run an order of magnitude faster); and prediction accuracy.
- Resilient back propagation achieved the best performance among the neural networks in terms of accuracy (97.04 %) and training (67 epochs).
- LGPs outperform SVMs and RBP in terms of detection accuracies with the expense of time

Regarding feature ranking, we observe that

- The two feature ranking methods produce largely consistent results: except for the class 1 (Normal) and class 4 (U2Su) data, the features ranked as Important by the two methods heavily overlap.
- The most important features for the two classes of 'Normal' and 'DOS' heavily overlap.
- 'U2Su' and 'R2L', the two smallest classes representing the most serious attacks, each has a small number of important features and a large number of secondary features.
- The performances of (a) using the important features for each class, Table 2, (b) using the union of important features, Table 4 and Table 5, and (c) using the union of important and secondary features for each class, Table 3, do not show significant differences, and are all similar to that of using all 41 features.
- Using the important features for each class gives the most remarkable performance: the testing time decreases in each class; the accuracy increases slightly for one class 'Normal', decreases slightly for two classes 'Probe' and 'DOS', and remains the same for the two most serious attack classes.

References

[1] Kendall K. (1998) "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", *Master's Thesis, Massachusetts Institute of Technology*.

[2] Hertz J., Krogh A., Palmer R. G. (1991) "Introduction to the Theory of Neural Computation," *Addison –Wesley.*

[3] Joachims T. (1998) "*Making Large-Scale SVM Learning Practical,"* LS8-Report, University of Dortmund, LS VIII-Report.

[4] Friedman J. H. (1991) "Multivariate Adaptive Regression Splines", Analysis of Statistics, Vol 19, pp. 1-141.

[5] Banzhaf. W., Nordin. P., Keller. E. R., and Francone F. D. (1998) "Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Applications," Morgan Kaufmann Publishers, Inc.

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

[7] Kumar S., Spafford E. H. (1994) "An Application of Pattern Matching in Intrusion Detection," *Technical Report CSD-TR-94-013*. Purdue University.

[8] Lunt T., Tamaru A., Gilham F., Jagannathan R., Jalali C., Neumann, P. G., Javitz, H. S., Valdes A., Garvey T. D. (1992) "*A real time Intrusion Detection Expert System (IDES) - Final Report*, "SRI International, Menlo Park, CA.

[9] Ilgun and Koral. (1993) "USTAT: A Real-time Intrusion Detection System for UNIX," *Proceedings of the 1993 Computer Society Symposium on Research in Security and Privacy*. Oakland, California, May 24-26, 1993. Los Alamitos, CA: IEEE Computer Society Press, pp. 16-29.

[10] Anderson D., Lunt T. F., Javitz H., Tamaru, Valdes A. (1995) "A. *Detecting Unusual Program Behavior Using the Stastistical Component of the Next-generation Intrusion Detection Expert System (NIDES)*," SRI-CSL-95-06, SRI International, Menlo Park, CA.

[11] Porras A. and Neumann P. G. EMERALD. (1997) "Event Monitoring Enabling Responses to Anomalous Live Disturbances," In *Proceedings of the National Information Systems Security Conference*, pp. 353-365.

[12] Debar H., Becke B., Siboni D. (1992) "A Neural Network Component for an Intrusion Detection System," Proceedings *of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 240-250.

[13] Cannady J. (1998) "Artificial Neural Networks for Misuse Detection," *National Information Systems Security Conference*, pp.368-381.

[14] S Mukkamala, G Janowski, A H. Sung. (2001) Intrusion Detection Using Neural Networks and Support Vector Machines. *Proceedings of Hybrid Information Systems* Advances in Soft Computing, Physica Verlag, Springer Verlag, ISBN 3790814806, pp.121-138.

[15] Stolfo J., Fan W., Lee W., Prodromidis A., and Chan P. K. (2000) "Cost-based Modeling and Evaluation for Data Mining With Application to Fraud and Intrusion Detection," *Results from the JAM Project by Salvatore.*

[16] Jianxiong L., and Bridges S. M. (2000) "Mining Fuzzy Association Rules and Fuzzy Frequency Episodes for Intrusion Detection," *International Journal of Intelligent Systems*, Vol. 15, No. 8, pp. 687-704

[17] Mukkamala S., and Sung A. H. (2003) Feature Selection for Intrusion Detection Using Neural Networks and Support Vector Machines. Journal of the Transportation Research Board of the National Academics, Transportation Research Record No 1822, pp. 33-39.

[18] Crosbie M., and Spafford E. H. (1995) "Defending a Computer System Using Autonomous Agents," *Technical Report CSD-TR-95-022.*

[19] Prodromidis L., and Stolfo S. J. (1999) "Agent-Based Distributed Learning Applied to Fraud Detection," *Technical Report*, CUCS-014-99.

[20] Dasgupta D. (1999) "Immunity –Based Intrusion Detection System: A general Framework," *Proceedings of* 22nd *National Information Systems Security Conference (NISSC)*, pp. 147-160.

[21] Helmer G., Wong J., Honavar V., and Miller L. (2003). Lightweight Agents for Intrusion Detection. *Journal of Systems and Software*, pp. 109-122.

[22] Kumar S., and Spafford E. H. (1994) "A pattern matching model for misuse intrusion detection," In *Proceedings of the 17th National Computer Security Conference*, pp. 11-21.

[23] Mahoney M., and Chan P. K. (2003) "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," *6th Intl. Symp. Recent Advances in Intrusion Detection*, pp. 220-237.

[24] Chan P. K., Mahoney M., and M. Arshad. (2003) "Learning Rules and Clusters for Anomaly Detection in Network Traffic," *Managing Cyber Threats: Issues, Approaches and Challenges* Kluwer, to appear.

[25] Frawley W., Piatetsky-Shapiro G., and C. Matheus C. (1992) "Knowledge Discovery in Databases: An Overview," AI Magazine, pp. 213-228.

[26] Lee W., and Stolfo S. J. (2000) A Framework for Constructing Features and Models for Intrusion Detection Systems. ACM Transactions on Information and System Security, Volume 3, Number 4, pp. 227-261.

[27] Zadeh L. A., Roles of Soft Computing and Fuzzy Logic in the Conception, Design and Deployment of Information/Intelligent Systems, Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, O. Kaynak, Zadeh L. A., Turksen B., Rudas I. J. (Eds.), pp 1-9.

[28] Riedmiller M., and Braun H. (1993) "A direct adaptive method for faster back propagation learning: The RPROP algorithm," *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco.

[29] Joachims T. (2000) "SVMlight is an Implementation of Support Vector Machines (SVMs) in C," http://ais.gmd.de/~thorsten/svm_light. University of Dortmund. Collaborative *Research Center on Complexity Reduction in Multivariate Data (SFB475)*.

[30] Vapnik V. (1995) "The Nature of Statistical Learning Theory. Springer-Verlag," New York.

[31] Steinberg D., Colla P. L., and Kerry M. (1999) "MARS User Guide", San Diego, CA: Salford Systems.