Feature deduction and ensemble design of intrusion detection systems

Srilatha Chebrolu[a], Ajith Abraham[a,b,*], Johnson P. Thomas[a]

[a]Department of Computer Science, Oklahoma State University, 700 N Greenwood Avenue, Tulsa, OK 74106, USA
[b]School of Computer Science and Engineering, Chung-Ang University, Seoul 156-756, Republic of Korea

Abstract Current intrusion detection systems (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. The purpose of this study is to identify important input features in building an IDS that is computationally efficient and effective. We investigated the performance of two feature selection algorithms involving Bayesian networks (BN) and Classification and Regression Trees (CART) and an ensemble of BN and CART. Empirical results indicate that significant input feature selection is important to design an IDS that is lightweight, efficient and effective for real world detection systems. Finally, we propose an hybrid architecture for combining different feature selection algorithms for real world intrusion detection.

Introduction to intrusion detection systems

Intrusion detection systems (IDS) were proposed to complement prevention-based security measures. An intrusion is defined to be a violation of the security policy of the system; intrusion detection thus refers to the mechanisms that are developed to detect violations of system security policy. Intrusion detection is based on the assumption that intrusive activities are noticeably different from normal system activities and thus detectable. Intrusion detection is not introduced to replace prevention-based techniques such as authentication and access control; instead, it is intended to complement existing security measures and detect actions that bypass the security monitoring and control component of the system. Intrusion

detection is therefore considered as a second line of defense for computer and network systems. Generally, an intrusion would cause loss of integrity, confidentiality, denial of resources, or unauthorized use of resources. Some specific examples of intrusions that concern system administrators include (Bishop, 2003):

- Unauthorized modifications of system files so as to facilitate illegal access to either system or user information.
- Unauthorized access or modification of user files or information.
- Unauthorized modifications of tables or other system information in network components (e.g. modifications of router tables in an internet to deny use of the network).
- Unauthorized use of computing resources (perhaps through the creation of unauthorized accounts or perhaps through the unauthorized use of existing accounts).

Some of the important features an intrusion detection system should possess include:

- Be fault tolerant and run continually with minimal human supervision. The IDS must be able to recover from system crashes, either accidental or caused by malicious activity.
- Possess the ability to resist subversion so that an attacker cannot disable or modify the IDS easily. Furthermore, the IDS must be able to detect any modifications forced on the IDS by an attacker.
- Impose minimal overhead on the system to avoid interfering with the normal operation of the system.
- Be configurable so as to accurately implement the security policies of the systems that are being monitored. The IDS must be adaptable to changes in system and user behavior over time.
- Be easy to deploy: this can be achieved through portability to different architectures and operating systems, through simple installation mechanisms, and by being easy to use by the operator.
- Be general enough to detect different types of attacks and must not recognize any legitimate activity as an attack (false positives). At the same time, the IDS must not fail to recognize any real attacks (false negatives).

An IDS maybe be a combination of software and hardware. Most IDSs try to perform their task in real time. However, there are also IDSs that do not operate in real time, either because of the nature of the analysis they perform or because they are meant for forensic analysis (analysis of what happened in the past to a system). There are some intrusion detection systems that try to react when they detect an unauthorized action. This reaction usually includes trying to limit the damage, for example by terminating a network connection.

Since the amount of audit data that an IDS needs to examine is very large even for a small network, analysis is difficult even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns (Lee et al., 1999a). Audit data capture various features of the connections. For example, the audit data would show the source and destination bytes of a TCP connection, or the number of failed login attempts or duration of a connection. Complex relationships exist between the features, which are difficult for humans to discover. An IDS must therefore reduce the amount of data to be processed. This is very important if real-time detection is desired. Some data may not be useful to the IDS and thus can be eliminated before processing. In complex classification domains, 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 have an impact on the accuracy of the IDS. Feature selection improves classification by searching for the subset of features, which best classifies the training data (Sung and Mukkamala, 2003).

In the literature several machine-learning paradigms, fuzzy inference systems and expert systems, have been used to develop IDSs (Lee et al., 1999a; Luo and Bridges, 2000). Sung and Mukkamala (2003) have demonstrated that a large number of features are unimportant and may be eliminated, without significantly lowering the performance of the IDS. The literature indicates very little scientific efforts aimed at modeling efficient IDS feature selection. The task of an IDS is often modeled as a classification problem in a machine-learning context. In this paper we investigate different data mining techniques for selecting a subset of significant features from a feature set for network data. This reduced feature set is then employed in an ensemble design to implement an IDS. Our approach results in faster real-time intrusion detection and more accurate detection.

In the next section the aims and objectives of different intrusion detection methods are presented. The different types of intrusion detection systems are described in Section 'Types of

intrusion detection systems'. In Section 'Data mining approaches toward intrusion detection' we review modern data mining approaches for intrusion detection. Our proposed data mining paradigms for intrusion detection, namely, for feature selection and classification based on Bayesian networks and Classification and Regression Tress (CART) are presented in fifth and sixth sections. Experimental results are reported in seventh section followed by conclusions.

Intrusion detection methods

The signatures of some attacks are known, whereas other attacks only reflect some deviation from normal patterns. Consequently, two main approaches have been devised to detect intruders.

Anomaly detection

Anomaly detection assumes that an intrusion will always reflect some deviations from normal patterns. Anomaly detection may be divided into static and dynamic anomaly detection. A static anomaly detector is based on the assumption that there is a portion of the system being monitored that does not change. Usually, static detectors only address the software portion of a system and are based on the assumption that the hardware need not be checked. The static portion of a system is the code for the system and the constant portion of data upon which the correct functioning of the system depends. For example, the operating systems' software and data to bootstrap a computer never change. If the static portion of the system ever deviates from its original form, an error has occurred or an intruder has altered the static portion of the system. Therefore static anomaly detectors focus on integrity checking (Forrest et al., 1994; Kim and Spafford, 1995). Dynamic anomaly detection typically operates on audit records or on monitored networked traffic data. Audit records of operating systems do not record all events; they only record events of interest. Therefore only behavior that results in an event that is recorded in the audit will be observed and these events may occur in a sequence. In distributed systems, partial ordering of events is sufficient for detection. In other cases, the order is not directly represented; only cumulative information, such as cumulative processor resource used during a time interval, is maintained. In this case, thresholds are defined to separate normal resource consumption from anomalous resource consumption.

Misuse detection

Misuse detection is based on the knowledge of system vulnerabilities and known attack patterns. Misuse detection is concerned with finding intruders who are attempting to break into a system by exploiting some known vulnerability. Ideally, a system security administrator should be aware of all the known vulnerabilities and eliminate them. The term intrusion scenario is used as a description of a known kind of intrusion; it is a sequence of events that would result in an intrusion without some outside preventive intervention. An intrusion detection system continually compares recent activity to known intrusion scenarios to ensure that one or more attackers are not attempting to exploit known vulnerabilities. To perform this, each intrusion scenario must be described or modeled. The main difference between the misuse techniques is in how they describe or model the behavior that constitutes an intrusion. The original misuse detection systems used rules to describe events indicative of intrusive actions that a security administrator looked for within the system. Large numbers of rules can be difficult to interpret. *If-then* rules are not grouped by intrusion scenarios and therefore making modifications to the rule set can be difficult as the affected rules are spread out across the rule set. To overcome these difficulties, new rule organizational techniques include model-based rule organization and state transition diagrams. Misuse detection systems use the rules to look for events that possibly fit an intrusion scenario. The events may be monitored live by monitoring system calls or later using audit records.

Advantages and disadvantages of anomaly detection and misuse detection

The main disadvantage of misuse detection approaches is that they will detect only the attacks for which they are trained to detect. Novel attacks or unknown attacks or even variants of common attacks often go undetected. At a time when new security vulnerabilities in software are discovered and exploited every day, the reactive approach embodied by misuse detection methods is not feasible for defeating malicious attacks. The main advantage of anomaly detection approaches is the ability to detect novel attacks or unknown attacks against software systems, variants of known attacks, and deviations of normal usage of programs regardless of whether the source is a privileged internal user or an unauthorized external user. The

disadvantage of the anomaly detection approach is that well-known attacks may not be detected, particularly if they fit the established profile of the user. Once detected, it is often difficult to characterize the nature of the attack for forensic purposes. Another drawback of many anomaly detection approaches is that a malicious user who knows that he or she is being profiled can change the profile slowly over time to essentially train the anomaly detection system to learn the attacker's malicious behavior as normal. Finally a high false positive rate may result for a narrowly trained detection algorithm, or conversely, a high false negative rate may result for a broadly trained anomaly detection approach.

Types of intrusion detection systems

There are two types of intrusion detection systems that employ one or both of the intrusion detection methods outlined above. Host-based systems base their decisions on information obtained from a single host (usually audit trails), while network-based intrusion detection systems obtain data by monitoring the traffic in the network to which the hosts are connected.

Host-based intrusion detection

A generic intrusion detection model proposed by Denning (1987) is a rule-based pattern matching system in which the intrusion detection tasks are conducted by checking the similarity between the current audit record and the corresponding profiles. If the current audit record deviates from the normal patterns, it will be considered an anomaly. Several IDSs were developed using profile and rule-based approaches to identify intrusive activity (Lunt et al., 1988).

Network-based intrusion detection

With the proliferation of computer networks, more and more individual hosts are connected into local area networks and/or wide area networks. However, the hosts, as well as the networks, are exposed to intrusions due to the vulnerabilities of network devices and network protocols. The TCP/IP protocol can be also exploited by network intrusions such as IP spoofing, port scanning, and so on. Therefore, network-based intrusion detection has become important and is designed to protect a computer network as well as all of its hosts. The installation of a network-based intrusion detection system can also decrease the burden of the intrusion detection task on every individual host.

Data mining approaches toward intrusion detection

In this paper we propose data mining approaches for intrusion detection. A review of intrusion detection systems that employ non-data mining techniques is therefore not presented. Data mining approaches are new methods in intrusion detection systems. Data mining is defined as the semi-automatic discovery of patterns, associations, changes, anomalies, rules, and statistically significant structures and events in data (Hand et al., 2001). Data mining attempts to extract knowledge in the form of models from data, which may not be seen easily with the naked eye. There exist many different types of data mining algorithms including classification, regression, clustering, association rule abduction, deviation analysis, sequence analysis etc.

Various data mining techniques have been applied to intrusion detection because it has the advantage of discovering useful knowledge that describes a user's or program's behavior from large audit data sets. Data mining has been used for anomaly detection (Lee and Stolfo, 1998; Lunt et al., 1992). Statistics (Anderson et al., 1995; Debar et al., 1992), Artificial Neural Network (ANN) (Cho and Park, 2003; Lippmann and Cunningham, 2000) and Hidden Markov Model (HMM) (Cohen, 1995), Rule Learning (Lazarevic et al., 2003), Outlier Detection scheme (Han and Cho, 2003), Support Vector Machines (Abraham, 2001), Neuro-Fuzzy (NF) computing (Mukkamala et al., 2003), Multivariate Adaptive Regression Splines (Banzhaf et al., 1998) and Linear Genetic Programming (Mukkamala et al., 2004b) are the main data mining techniques widely used for anomaly and misuse detections.

Statistics is the most widely used technique, which defines normal behavior by collecting data relating to the behavior of legitimate users over a period of time (Anderson et al., 1995). Next-generation Intrusion Detection Expert Systems (NIDES) is the representative IDS based on statistics that measures the similarity between a subject's long-term behavior and short term behavior for intrusion detection (Debar et al., 1992). The detection rate is high because it can use various types of audit data and detects intrusion based on the previous experimental data. In NIDES known

attacks and intrusion scenarios are encoded in a rule base. It is therefore not sensitive to some behaviors and detectable types of intrusions are limited. Hyperview is a representative IDS using neural networks (Lippmann and Cunningham, 2000). It consists of two modules: a neural network and an expert system. Lippmann and Cunningham (2000) have applied neural networks to a keyword-based detection system. While the artificial neural network has some similarity to statistical techniques, it has the advantage of easier representation of nonlinear relationships between input and output. Even if the data were incomplete or distorted, a neural network would be capable of analyzing the data from a network. Another advantage of neural networks is its inherent computational speed. The defects of neural networks are that its computational load is very heavy and it is difficult to interpret the relationship between inputs and outputs. An HMM is a useful tool to model the sequence of observed symbols of which the construction mechanism cannot be known (Cohen, 1995). While HMM produces better performance in modeling system call events compared to other methods, it requires a very long time for modeling normal behaviors. Using this model, raw data are first converted into ASCII network packet information, which in turn is converted into connection level information using Mining Audit Data for Automated Models for Intrusion Detection (MADAMID) (Lee et al., 1999b). RIPPER (Lazarevic et al., 2003), a rule learning tool, is then applied to the data generated by MADAMID. RIPPER automatically mines the patterns of intrusion. Although it is a good tool for discovering known patterns, an anomaly detection technique is required for the detection of novel intrusions. Another data mining technique, the outlier detection scheme attempts to identify a data point that is very different from the rest of the data. Lazarevic et al. (2003) have applied it to anomaly detection. Support Vector Machines (SVM) are learning machines that plot training vectors in high-dimensional feature space, labeling each vector by its class. SVMs have proven to be a good candidate for intrusion detection because of its speed and scalability (Mukkamala et al., 2003).

In Neuro-Fuzzy (NF) computing (Abraham, 2001), if we have knowledge expressed in the form of linguistic rules, we can build a Fuzzy Inference System (FIS), and if we have data, then we can use ANNs. While the learning capability is an advantage from the viewpoint of FIS, the formation of a linguistic rule base will be advantageous from the viewpoint of ANN. An Adaptive Neuro-Fuzzy IDS is proposed in (Shah et al., 2004). Multivariate Adaptive Regression Splines (MARS) is an innovative approach that automates the building of accurate predictive models for continuous and binary dependent variables. It excels at finding optimal variable transformations and interactions, and the complex data structure that often hides in high-dimensional data (Abraham and Steinberg, 2001; Banzhaf et al., 1998). An IDS based on MARS is proposed in (Abraham and Steinberg, 2001). In Linear Genetic Programming (LGP) (as opposed to tree-based Genetic Programming (GP)) (Sequeira and Zaki, 2002) computer programs are evolved at the machine code level, using lower level representations for the individuals. This can tremendously hasten up the evolution process. LGP based IDS is presented in (Mukkamala et al., 2004a). To overcome the drawbacks of single-measure detectors, a multiple measure intrusion detection method is proposed in (Jolliffe, 1986). In this approach hidden Markov model, statistical method and rule-based method are integrated with a rule-based approach. Chebrolu et al. (2004) have proposed an ensemble IDS that combines the strengths of Bayesian networks and Classification and Regression Trees for intrusion detection.

None of the above works have proposed a rigorous and scientific approach for increasing and improving the efficiency of intrusion detection. The next section outlines our proposed data mining approach for faster and more effective intrusion detection.

The data mining process of building intrusion detection models

Raw (binary) audit data are first processed into ASCII network packet information (or host event data), which is in turn summarized into connection records (or host session records) containing a number of within-connection features, e.g., service, duration, flag etc. (indicating the normal or error status according to the protocols). Data mining programs are then applied to the connection records to compute the frequent patterns i.e. association rules and frequent episodes, which are in turn analyzed to construct additional features for the connection records. Classification algorithms are then used to inductively learn the detection model. This process is of course iterative. For example, poor performance of the classification models often indicates that more pattern mining and feature construction is needed.

Importance of data reduction for intrusion detection systems

IDSs have become important and widely used tools for ensuring network security. Since the amount of audit data that an IDS needs to examine is very large even for a small network, classification by hand is impossible. Analysis is difficult 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 are 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.

Data filtering
The purpose of data filtering is to reduce the amount of data directly processed by the IDS. Some data may not be useful to the IDS and thus can be eliminated before processing. This has the advantage of decreasing storage requirements, reducing processing time and improving the detection rate (as data irrelevant to intrusion detection are discarded). However, filtering may throw out useful data, and so must be done carefully.

Feature selection
In complex classification domains, some data may hinder the classification process. 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.

Data clustering
Clustering can be performed to find hidden patterns in data and significant features for use in detection. Clustering can also be used as a reduction technique by storing the characteristics of the clusters instead of the individual data. In previous work a number of experiments have been performed to measure the performance of different machine-learning paradigms as mentioned in the previous section. Classifications were performed on the binary (normal/attack) as well as five-class classifications (normal, and four classes of attacks). It has been demonstrated that a large number of the (41) input features are unimportant and may be eliminated, without significantly lowering the performance of the IDS (Sung and Mukkamala, 2003). In terms of the five-class classification, Sung and Mukkamala (2003) found that by using only 19 of the most important features, instead of the entire 41-feature set, the change in accuracy of intrusion detection was statistically insignificant. Sung and Mukkamala (2003) applied the technique of deleting one feature at time. Each reduced feature set was then tested on Support Vector Machines and Neural Networks to rank the importance of input features. The reduced feature set that yielded the best detection rate in the experiments was considered to be the set of important features.

Unlike the work reported in (Sung and Mukkamala, 2003) which employed a trial-and-error based approach, we investigate feature reduction using data mining techniques. Our work correspondingly focuses on approaches that will improve the performance of IDSs by providing real-time intrusion detection. This is achieved by reducing the data space and then classifying intrusions based on the reduced feature space. We use data mining techniques including Bayesian networks and Classification and Regression Trees (CART). Bayesian networks not only classify the data, but also select features based on the Markov blanket of the target variables. CART classifies data by constructing a decision tree. Furthermore, the CART algorithm automatically produces a predictor ranking (variable importance) based on the contribution predictors make to the construction of

the decision tree, thus helping to identify which features are important for intrusion detection.

Feature selection and classification using AI paradigms

A novel aspect of our work is the use of data mining techniques to select significant features for intrusion detection. In particular we investigate Bayesian networks and the CART algorithm for this purpose.

Bayesian learning and Markov blanket modeling of input features

The Bayesian network (BN) is a powerful knowledge representation and reasoning algorithm under conditions of uncertainty. A Bayesian network $B = (N, A, \Theta)$ is a Directed Acyclic Graph (DAG) (N, A) where each node $n \in N$ represents a domain variable (e.g. a data set 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. A BN can be used to compute the conditional probability of one node, given values assigned to the other nodes. Many Bayesian network structure-learning algorithms have been developed. These algorithms generally fall into two groups, search and scoring based algorithms and dependency analysis based algorithms. Although some of these algorithms can give good results on some benchmark data sets, there are still several problems such as node ordering requirement, lack of efficiency and lack of publicly available learning tools (Neapolitan, 1990). In order to resolve these problems, two types of algorithms have been developed in the area of Bayesian network structure learning. Type 1 deals with a special case where the node ordering is given, which requires $O(N^2)$ Conditional Independence (CI) tests and is correct given that the underlying model is DAG faithful. Type 2 deals with the general case and requires $O(N^4)$ CI tests and is correct given that the underlying model is monotone DAG faithful.

The major advantage of Bayesian networks over many other types of predictive models, such as neural networks and decision trees, is that unlike those "black box" approaches, the Bayesian network structure represents the inter-relationships among the data set attributes. Human experts can easily understand the network structures and if necessary can easily modify them to obtain better predictive models. By adding decision nodes and utility nodes, BN models can also be extended to decision networks for decision analysis. Other advantages of Bayesian networks include explicit uncertainty characterization, fast and efficient computation, and quick training. They are highly adaptive and easy to build, and provide explicit representation of domain specific knowledge in human reasoning frameworks. Moreover, Bayesian networks offer good generalization with limited training data and easy maintenance when adding new features or new training data.

One of the main objectives of this work is to select features for classification. Given the following:

(i) a set of features F (including an output or target variable T) instantiated by some process P;
(ii) a classification algorithm $A(D_{tr}, T, F_i) \rightarrow a_i$ (where D_{tr} is any set of training instances sampled from P, a_i is the model or class of models returned by $A()$ for some subset F_i of F);
(iii) a classification performance metric M, defined over the space of the outputs of $A()$ and the space of test sets D_{te}.

The feature selection problem for classification may be defined as: find a minimal subset of F that maximizes performance of $A()$ according to M, given the above.

Markov blanket (MB) of the output variable T, is a novel idea for significant feature selection in large data sets (Tsamardinos et al., 2003). 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 (Cheng et al., 2002). The formal definition (Dzeroski and Zenko, 2002) 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

Table 1   Network data feature labels

Label	Network data feature	Label	Network data feature	Label	Network data feature	Label	Network data feature
A	duration	L	logged_in	W	count	AH	dst_host_same_srv_rate
B	protocol-type	M	num_compromised	X	srv_count	AI	dst_host_diff_srv_rate
C	service	N	root_shell	Y	serror_rate	AJ	dst_host_same_src_port_rate
D	flag	O	su_attempted	Z	srv_serror_rate	AK	dst_host_srv_diff_host_rate
E	src_bytes	P	num_root	AA	rerror_rate	AL	dst_host_serror_rate
F	dst_bytes	Q	num_file_creations	AB	srv_rerror_rate	AM	dst_host_srv_serror_rate
G	land	R	num_shells	AC	same_srv_rate	AN	dst_host_rerror_rate
H	wrong_fragment	S	num_access_files	AD	diff_srv_rate	AO	dst_host_srv_rerror_rate
I	urgent	T	num_outbound_cmds	AE	srv_diff_host_rate		
J	hot	U	is_host_login	AF	dst_host_count		
K	num_falied_logins	V	is_guest_login	AG	dst_host_srv_count		

data, the Markov blanket of the class node forms feature selection and all features outside the Markov blanket are deleted from the BN.

CART learning and modeling

The Classification and Regression Trees (CART) methodology is based on binary recursive partitioning (Brieman et al., 1984). The process is binary because parent nodes are always split into exactly two child nodes and recursive because the process is repeated by treating each child node as a parent. The key elements of CART analysis are a set of rules for splitting each node in a tree; deciding when the tree is complete and assigning a class outcome to each terminal node. As an example, for the DARPA intrusion data set (KDD cup 99 intrusion detection data set) with 5092 cases and 41 variables, CART considers up to 5092 times 41 splits for a total of 208,772 possible splits. For splitting, the Gini rule is used which essentially is a measure of how well the splitting rule separates the classes contained in the parent node. Splitting is impossible if only one case remains in a particular node or if all the cases in that node are exact copies of each other or if a node has too few cases. Instead of attempting to decide whether a given node is terminal or not, the algorithm proceeds by growing trees until it is not possible to grow them any further. Once the algorithm has generated a maximal tree, it examines smaller trees obtained by pruning away branches of the maximal tree. Unlike other methods, CART does not stop in the middle of the tree-growing process, because there might still be important information to be discovered by drilling down several more levels. Once the maximal tree is grown and a set of sub-trees is derived from it, CART determines the best tree by testing for error rates or costs. The misclassification error rate is calculated for the largest tree and also for every

sub-tree. The best sub-tree is the one with the lowest or near-lowest cost, which may be a relatively small tree.

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 nth surrogate instead of the primary variable, then the importance becomes $i_{importance} = (p^n) \times i_{improvement}$ in which p is the 'surrogate improvement weight' which is a user controlled parameter set between 0 and 1.

Experiment setup and results

The data for our experiments were prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Laboratory. The data set contains 24 attack types that could be classified into four main categories namely *Denial of Service (DOS), Remote to User (R2L), User to Root (U2R)* and *Probing*. The original data contain 744 MB data with 4,940,000 records. The data set has 41 attributes for each connection record plus one

Table 2 Performance of Bayesian belief network

Attack class	41 variables			17 variables		
	Train (s)	Test (s)	Accuracy (%)	Train (s)	Test (s)	Accuracy (%)
Normal	42.14	19.02	99.57	23.29	11.16	*99.64*
Probe	49.15	21.04	*99.43*	25.07	13.04	98.57
DOS	54.52	23.02	99.69	28.49	14.14	98.16
U2R	30.02	15.23	*64.00*	14.13	7.49	60.00
R2L	47.28	12.11	*99.11*	21.13	13.57	98.93

class label. Some features are derived features, which are useful in distinguishing normal connection from attacks. These features are either nominal or numeric. Some features examine only the connections in the last 2 s that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. These are called same host features. Some features examine only the connections in the past 2 s that have the same service as the current connection and are called same service features. Some other connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. These are called host-based traffic features. R2L and U2R attacks do not have any sequential patterns like DOS and Probe because the former attacks have the attacks embedded in the data packets whereas the later attacks have many connections in a short amount of time. So some features that look for suspicious behavior in the data packets like number of failed logins are constructed and these are called content features. Our experiments have three phases namely data reduction, a training phase and a testing phase. In the data reduction phase, important variables for real-time intrusion detection are selected by feature selection. In the training phase, the Bayesian neural network and Classification and Regression Trees construct a model using the training data to give maximum generalization accuracy on the unseen data. The test data are then passed through the saved trained model to detect intrusions in the testing phase. The data set for our experiments contains randomly generated 11,982 records having 41 features (KDD cup 99 intrusion detection data set). The labels of the 41 features and their corresponding network data features are shown in Table 1.

This data set has five different classes namely *Normal*, *DOS*, *R2L*, *U2R* and *Probes*. The training and test comprises of 5092 and 6890 records, respectively. All the IDS models were trained and tested with the same set of data. As the data set has five different classes we performed a five-class binary classification. The *Normal* data belongs to class 1, *Probe* belongs to class 2, *DOS* belongs to class 3, *U2R* belongs to class 4 and *R2L* belongs to class 5. All experiments were performed using an AMD Athlon 1.67 GHz processor with 992 MB of RAM.

Modeling IDS using Bayesian network

We selected the important features using the Markov blanket model and found that out of the 41-variables 17-variables of the data set form the Markov blanket of the class node as explained in Section 'Importance of data reduction for intrusion detection systems'. These 17-variables are *A, B, C, E, G, H, K, L, N, Q, V, W, X, Y, Z, AD* and *AF*. Furthermore, a Bayesian network classifier was constructed using the training data and then the classifier was used on the test data set to classify the data as an attack or normal data.

Table 3 Performance of classification and regression trees

Attack class	41-variable data set			12-variable data set		
	Train (s)	Test (s)	Accuracy (%)	Train (s)	Test (s)	Accuracy (%)
Normal	1.15	0.18	99.64	0.80	0.02	*100.00*
Probe	1.25	0.03	*97.85*	0.85	0.05	97.71
DOS	2.32	0.05	*99.47*	0.97	0.07	85.34
U2R	1.10	0.02	48.00	0.45	0.03	*64.00*
R2L	1.56	0.03	90.58	0.79	0.02	*95.56*

Table 4   Performance of Bayesian and CART using reduced data sets

Attack class	Bayesian with 12 variables			CART with 17 variables		
	Train (s)	Test (s)	Accuracy (%)	Train (s)	Test (s)	Accuracy (%)
Normal	20.10	10.13	98.78	1.03	0.04	99.64
Probe	23.15	11.17	99.57	1.15	0.13	100.00
DOS	25.19	12.10	98.95	0.96	0.11	99.97
U2R	11.03	5.01	48.00	0.59	0.02	72.00
R2L	19.05	12.13	98.93	0.93	0.10	96.62

Table 2 depicts the performance of the Bayesian belief network by using the original 41-variable data set and the 17-variable reduced data set. The training and testing times for each classifier decreases when the 17-variable data set is used. For R2L class the test time was a bit longer for the 17-variable data set. Using the 17-variable data set there was a slight increase in the performance accuracy for the *Normal* class compared to the 41-variable data set.

Modeling IDS using Classification and Regression Trees

The important variables for intrusion detection were 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 as explained in the previous section 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. Further the classifier was constructed using the training data and then the test data were passed through the saved trained model. Table 3 compares the performance of CART using the 41-variable original data set and the 12-variable reduced data set. Normal class is classified 100% correctly. Furthermore, the accuracies of classes U2R and R2L have increased by using the 12-variable reduced data set. It is observed that CART classifies accurately on smaller data sets.
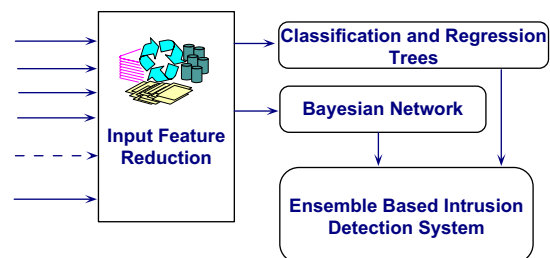
Furthermore, we used the Bayesian reduced 17-variable data set (Section 'Host-based intrusion detection') to train CART and the CART reduced 12-variable data set (Section 'Network-based intrusion detection') to train the Bayesian network. As illustrated in Table 4 except for R2L all the other classes were classified accurately by the CART algorithm. Moreover, the training and testing times for each class was greater for the Bayesian network classifier compared to the CART algorithm.

Performance using reduced data set of (Sung and Mukkamala, 2003)

We also attempted to evaluate the performance of CART and the Bayesian network using the reduced data set (same input variables) given in (Sung and Mukkamala, 2003). Table 5 shows the performance comparisons of CART and the Bayesian network using 19 variables. Except for the *U2R* class, the 17 and 12-variable data sets performed well for all the other classes.

## Ensemble approach using reduced data sets

Several researchers have investigated the combination of different classifiers to form an ensemble classifier (Dzeroski and Zenko, 2002; Ji and Ma, 1997). An important advantage for combining

Table 5   Performance of CART and Bayesian network using 19 variables

Class	Bayesian	CART
Normal	99.57	95.50
Probe	96.71	96.85
DOS	99.02	94.31
U2R	56.00	84.00
R2L	97.87	97.69

Figure 1   Ensemble approach for IDS.

Table 6 Performance of ensemble approach using different data sets

Class	12 variables	17 variables	41 variables
Normal	*100.00*	99.64	99.71
Probe	99.86	*100.00*	99.85
DOS	99.98	*100.00*	99.93
U2R	*80.00*	72.00	72.00
R2L	*99.47*	99.29	99.47

redundant and complementary classifiers is to increase robustness, accuracy, and better overall generalization. In this approach we first constructed the Bayesian network classifier and the CART models individually to obtain a very good generalization performance. The ensemble approach was used for the 12, 17 and 41-variable data sets and is illustrated in Fig. 1. In the ensemble approach, the final outputs were decided as follows: each classifier's output is given a weight (0—1 scale) depending on the generalization accuracy as given in Tables 2—4. If both classifiers agree then the output is decided accordingly. If there is a conflict then the decision given by the classifier with the highest weight is taken into account. Table 6 illustrates the ensemble results using the different data sets. From the results, we conclude that the ensemble approach gives better performance than the two individuals separately used models. The ensemble approach basically exploits the differences in misclassification (by individual models) and improves the overall performance. Fig. 2 illustrates the developed hybrid IDS model after summarizing all the empirical results. By using the hybrid model *Normal*, *Probe* and *DOS* could be detected with 100% accuracy and *U2R* and *R2L* with 84% and 99.47% accuracies, respectively.
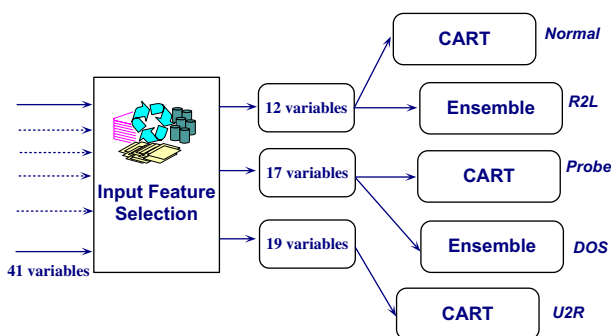
Figure 2 Developed ensemble IDS model for different attack classes.

Conclusions

In this research we have investigated new techniques for intrusion detection and performed data reduction and evaluated their performance on the DARPA benchmark intrusion data. Our initial experiments using Principal Component Analysis and Independent Component Analysis (Hyvärinen et al., 2001) (PCA/ICA) to compress data did not yield satisfactory feature reduction from an intrusion detection perspective. We used the feature selection method using Markov blanket model and decision tree analysis. Following this, we explored the general Bayesian network (BN) classifier and Classification and Regression Trees (CART) as intrusion detection models. We have also demonstrated performance comparisons using different reduced data sets. The proposed ensemble of BN and CART combines the complementary features of the base classifiers. Finally, we propose a hybrid architecture involving ensemble and base classifiers for intrusion detection. From the empirical results, it is seen that by using the hybrid model *Normal*, *Probe* and *DOS* could be detected with 100% accuracy and *U2R* and *R2L* with 84% and 99.47% accuracies, respectively. Our future research will be directed towards developing more accurate base classifiers particularly for the detection of *U2R* type of attacks.

Acknowledgements

References

Abraham Ajith. Neuro-fuzzy systems: state-of-the-art modeling techniques, connectionist models of neurons, learning processes, and artificial intelligence. In: Mira Jose, Prieto Alberto, editors. Lecture notes in computer science. vol. 2084. Germany: Springer-Verlag; 2001. p. 269—76. Granada, Spain.

Abraham Ajith, Steinberg Dan. MARS: still an alien planet in soft computing? In: Alexandrov Vassil N, et al, editor. Lecture notes in computer science 2074. Germany: Springer-Verlag; San Francisco; 2001. p. 235—44.

Anderson D, Lunt TF, Javits H, Tamaru A, Valdes A. Detecting unusual program behavior using the statistical components of NIDES. In: NIDES technical report. SRI International; May 1995.

Banzhaf W, Nordin P, Keller ER, Francone FD. Genetic programming: an introduction on the automatic evolution of

computer programs and its applications: Morgan Kaufmann Publishers, Inc.; 1998.

Bishop Matt. Computer security — art and science: Addison Wesley; 2003.

Brieman L, Friedman J, Olshen R, Stone C. Classification of regression trees. Wadsworth Inc.; 1984.

Chebrolu Srilatha, Abraham Ajith, Thomas Johnson. Hybrid feature selection for modeling intrusion detection systems. In: Pal NR, et al, editor. 11th International conference on neural information processing, ICONIP'04. Lecture Notes in Computer Science. vol. 3316. Germany: Springer Verlag; 2004. p. 1020—5. ISBN 3-540-23931-6.

Cheng J, Greiner R, Kelly J, Bell DA, Liu W. Learning Bayesian networks from data: an information-theory based approach. The Artificial Intelligence Journal 2002;137:43—90.

Cho S-B, Park H-J. Efficient anomaly detection by modeling privilege flows with hidden Markov model. Computers and Security 2003;22(1):45—55.

Cohen WW. Fast effective rule induction. In: Proceedings of the 12th international conference on machine learning. July 1995. p. 115—23.

Debar H, Becker M, Siboni D. A neural network component for an intrusion detection system. Proceedings of 1992 IEEE computer society symposium on research in security and privacy. Oakland, CA; May 1992. p. 240—50.

Denning DE. An intrusion-detection model. IEEE Transactions on Software Engineering 1987;SE-13(2):222—32.

Dzeroski S, Zenko B. Is combining classifiers better than selecting the best one. ICML 2002; 2002. p. 123—30.

Forrest S, Perelson AS, Allen L, Cherukuri R. Self-nonself discrimination in a computer. In: Proceedings of the 1994 IEEE symposium on research in security and privacy. Los Alamitos, CA: IEEE Computer Society Press; 1994.

Han Sang-Jun, Cho Sung-Bae. Detecting intrusion with rule-based integration of multiple models. Computers and Security October 2003;22(7):613—23.

Hand David J, Mannila Heikki, Smyth Padhraic. Principles of data mining (adaptive computation and machine learning): Bradford Books; 2001.

Hyvärinen A, Karhunen J, Oja E. Independent component analysis: John Wiley & Sons; 2001.

Ji C, Ma S. Combinations of weak classifiers. IEEE Transaction on Neural Networks 1997;8(1):32—42.

Jolliffe IT. Principal component analysis: Springer-Verlag; 1986.

KDD cup 99 intrusion detection data set, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz>.

Kim Gene H, Spafford Eugene H. Experiences with tripwire: using integrity checkers for intrusion detection, <http://citeseer.ist.psu.edu/kim95experiences.html>; 1995.

Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J. A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of Third SIAM Conference on Data Mining; May 2003.

Lee W, Stolfo S. Data mining approaches for intrusion detection. Proceedings of the seventh USENIX security symposium, San Antonio, Texas; January 26—29, 1998.

Lee W, Stolfo S, Mok K. A data mining framework for building intrusion detection models. Proceedings of the IEEE symposium on security and privacy; 1999a.

Lee W, Stolfo S, Mok K. A data mining framework for building intrusion detection models. In: Proceedings of the IEEE symposium on security and privacy; 1999b.

Lippmann R, Cunningham S. Improving intrusion detection performance using keyword selection and neural networks. Computer Networks 2000;34(4):594—603.

Lunt TF, Jagannathan R, Lee R, Listgarten S, Edwards DL, Javitz HS, et al. IDES: the enhanced prototype — a real-time intrusion-detection expert system Number SRI-CSL-88-12. Menlo Park, CA: Computer Science Laboratory, SRI International; 1988.

Lunt TF, Tamaru A, Gilham F, Jagannathan R, Jalali C, Neuman PG. A real-time intrusion-detection expert system (IDES). In: Technical report project 6784. Computer Science Laboratory, SRI International; February 1992.

Luo J, Bridges SM. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. International Journal of Intelligent Systems John Wiley & Sons 2000;15(8):687—704.

MIT Lincoln Laboratory, <http://www.ll.mit.edu/IST/ideval/>.

Mukkamala Srinivas, Sung Andrew H, Abraham Ajith. Intrusion detection using ensemble of soft computing paradigms. In: Third international conference on intelligent systems design and applications. Intelligent systems design and applications, advances in soft computing. Germany: Springer Verlag; 2003. p. 239—48.

Mukkamala Srinivas, Sung Andrew H, Abraham Ajith. Modeling intrusion detection systems using linear genetic programming approach. In: Orchard Robert, Yang Chunsheng, Ali Moonis, editors. The 17th international conference on industrial & engineering applications of artificial intelligence and expert systems. Innovations in applied artificial intelligence. Germany: Springer Verlag; 2004a. p. 633—42. LNCS 3029.

Mukkamala Srinivas, Sung Andrew H, Abraham Ajith, Ramos Vitorino. Intrusion detection systems using adaptive regression splines. In: Seruca I, Filipe J, Hammoudi S, Cordeiro J, editors. Sixth international conference on enterprise information systems. ICEIS'04, Portugal, vol. 3. 2004b. p. 26—33. ISBN 972-8865-00-7.

Neapolitan RE. Probabilistic reasoning in expert systems: theory and algorithms. John Wiley and Sons; 1990.

Shah Khusbu, Dave Neha, Chavan Sampada, Mukherjee Sanghamitra, Abraham Ajith, Sanyal Sugata. Adaptive neuro-fuzzy intrusion detection system. In: IEEE international conference on information technology: coding and computing (ITCC'04), Portugal, vol. 1. USA: IEEE Computer Society; 2004. p. 70—4.

Sequeira Karlton, Zaki Mohammed. ADMIT: anomaly based data mining for intrusions. Conference on knowledge discovery in data, Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. Edmonton, Alberta, Canada; 2002. p. 386—95. ISBN 1-58113-567-X.

Sung AH, Mukkamala S. Identifying important features for intrusion detection using support vector machines and neural networks. In: Proceedings of International Symposium on Applications and the Internet (SAINT 2003); 2003. p. 209—17.

Tsamardinos I, Aliferis CF, Statnikov A. Time and sample efficient discovery of Markov blankets and direct causal relations. In: Ninth ACM SIGKDD international conference on knowledge discovery and data mining. USA: ACM Press; 2003. p. 673—8.

Srilatha Chebrolu received Master of Science degree in Computer Science from Oklahoma State University. Her primary research interests are in Internet security and data mining. Currently she is engaged as a consultant in the software industry.

Ajith Abraham is currently a Distinguished Visiting Professor at Chung-Ang University Seoul, Korea. He received PhD degree from Monash University, Australia. His primary research interests are in computational intelligence with

application areas including information security, bioinformatics, Web intelligence, energy management, financial modeling, etc. He has co-authored over 100 research publications in peer reviewed reputed journals and international conference proceedings of which three have won "best paper" awards. Prior to joining Chung-Ang University, he was working with Oklahoma State University, USA and Monash University, Australia. Before turning into a full time academic, he has been working with three multi-national companies on several industrial research and development projects for nearly eight years.

He is the founding Co Editor-in-Chief of The International Journal of Hybrid Intelligent Systems (IJHIS), IOS Press, Netherlands, Editor of the International Journal of Systems Science (IJSS), Taylor & Francis Group, London and Editor of Journal of Digital and Information Management (JDIM), Digital Information Research Foundation, India. 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 was also the General Chair of the 9th Online World Conference on Soft Computing in Industrial Applications (WSC9) and the General Co-Chair of The Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST05). More information at http://ajith.softcomputing.net.

Johnson P. Thomas obtained his B.Sc. from the University of Wales, UK, M.Sc. from the University of Edinburgh, Scotland and Ph.D. from the University of Reading, England. He has worked as a lecturer at the University of Reading and as an Associate Professor at Pace University, New York. He is currently an Assistant Professor of Computer Science at Oklahoma State University, USA. His primary research interest is in Computer Security including network and sensor security. He is also interested in Geographical Information Systems (GIS).

Available online at www.sciencedirect.com

SCIENCE *d* DIRECT®