Available online at www.sciencedirect.com
ScienceDirect
Journal of Network and Computer Applications 30 (2007) 114–132
Journal of NETWORK and COMPUTER APPLICATIONS
www.elsevier.com/locate/jnca

Modeling intrusion detection system using hybrid intelligent systems

Sandhya Peddabachigari[a], Ajith Abraham[b,*], Crina Grosan[c], Johnson Thomas[a]

[a]*Computer Science Department, Oklahoma State University, OK 74106, USA*
[b]*School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea*
[c]*Department of Computer Science, Babes-Bolyai University, Cluj-Napoca 3400, Romania*

Received 28 June 2005; accepted 28 June 2005

Abstract

The process of monitoring the events occurring in a computer system or network and analyzing them for sign of intrusions is known as intrusion detection system (IDS). This paper presents two hybrid approaches for modeling IDS. Decision trees (DT) and support vector machines (SVM) are combined as a hierarchical hybrid intelligent system model (DT–SVM) and an ensemble approach combining the base classifiers. The hybrid intrusion detection model combines the individual base classifiers and other hybrid machine learning paradigms to maximize detection accuracy and minimize computational complexity. Empirical results illustrate that the proposed hybrid systems provide more accurate intrusion detection systems.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Intrusion detection system; Hybrid intelligent system; Decision trees; Support vector machines; Ensemble approach

*Corresponding author.
E-mail addresses: ajith.abraham@ieee.org (A. Abraham), crina.grosan@ieee.org (C. Grosan), jpt@cs.okstate.edu (J. Thomas).

1084-8045/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2005.06.003

1. Introduction

Traditional protection techniques such as user authentication, data encryption, avoiding programming errors and firewalls are used as the first line of defense for computer security. If a password is weak and is compromised, user authentication cannot prevent unauthorized use, firewalls are vulnerable to errors in configuration and suspect to ambiguous or undefined security policies (Summers, 1997). They are generally unable to protect against malicious mobile code, insider attacks and unsecured modems. Programming errors cannot be avoided as the complexity of the system and application software is evolving rapidly leaving behind some exploitable weaknesses. Consequently, computer systems are likely to remain unsecured for the foreseeable future. Therefore, intrusion detection is required as an additional wall for protecting systems despite the prevention techniques. Intrusion detection is useful not only in detecting successful intrusions, but also in monitoring attempts to break security, which provides important information for timely countermeasures (Heady et al., 1990; Sundaram, 1996). Intrusion detection is classified into two types: misuse intrusion detection and anomaly intrusion detection.

Misuse intrusion detection uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions (Kumar and Spafford, 1995). These patterns are encoded in advance and used to match against user behavior to detect intrusions. Anomaly intrusion detection identifies deviations from the normal usage behavior patterns to identify the intrusion. The normal usage patterns are constructed from the statistical measures of the system features, for example, the CPU and I/O activities by a particular user or program. The behavior of the user is observed and any deviation from the constructed normal behavior is detected as intrusion.

Several machine-learning paradigms including neural networks (Mukkamala et al., 2003), linear genetic programming (LGP) (Mukkamala et al., 2004a), support vector machines (SVM), Bayesian networks, multivariate adaptive regression splines (MARS) (Mukkamala et al., 2004b) fuzzy inference systems (FISs) (Shah et al., 2004), etc. have been investigated for the design of IDS. In this paper, we investigate and evaluate the performance of decision trees (DT), SVM, hybrid DT–SVM and an ensemble approach. The motivation for using the hybrid approach is to improve the accuracy of the intrusion detection system when compared to using individual approaches. The hybrid approach combines the best results from the different individual systems resulting in more accuracy. The rest of the paper is organized as follows. The Literature review is presented in Section 2 followed by a short theoretical background on the machine-learning paradigms used in this research. Experimental results and analysis is presented in Section 4 and conclusions presented at the end.

2. Related research

With the proliferation of networked computers and the Internet, their security has become a primary concern. In 1980, James Anderson proposed that audit trails should be used to monitor threats (Anderson, 1980). The importance of such data had not been comprehended at that time and all the available system security procedures were focused on denying access to sensitive data from an unauthorized source. Dorothy (Denning, 1997) proposed the concept of intrusion detection as a solution to the problem of providing a sense of security in computer systems. This intrusion detection model is independent of system, type of intrusion and application environment. The basic idea is that intrusion behavior involves abnormal usage of the system. This model is known as rule-based pattern matching system. Some models of normal usage of the system could be constructed and verified against usage of the system and any significant deviation from the normal usage is flagged as abnormal usage. This model served as abstract model for further developments in the field and is known as the generic intrusion detection model and is depicted in Fig. 1. Different techniques and approaches have been used in later developments. The main techniques used are statistical approaches, predictive pattern generation, expert systems, keystroke monitoring, model-based Intrusion detection, state transition analysis, pattern matching, and data mining techniques.

Statistical approaches compare the recent behavior of a user of a computer system with observed behavior and any significant deviation is considered as intrusion. This approach requires construction of a model for normal user behavior. Any user behavior that deviates significantly from this normal behavior is flagged as an intrusion. Intrusion detection expert system (IDES) (Lunt, 1993) exploited the statistical approach for the detection of intruders. It uses the intrusion detection model proposed by Denning (1997) and audit trails data as suggested in Anderson (1980). IDES maintains profiles, which is a description of a subject's normal

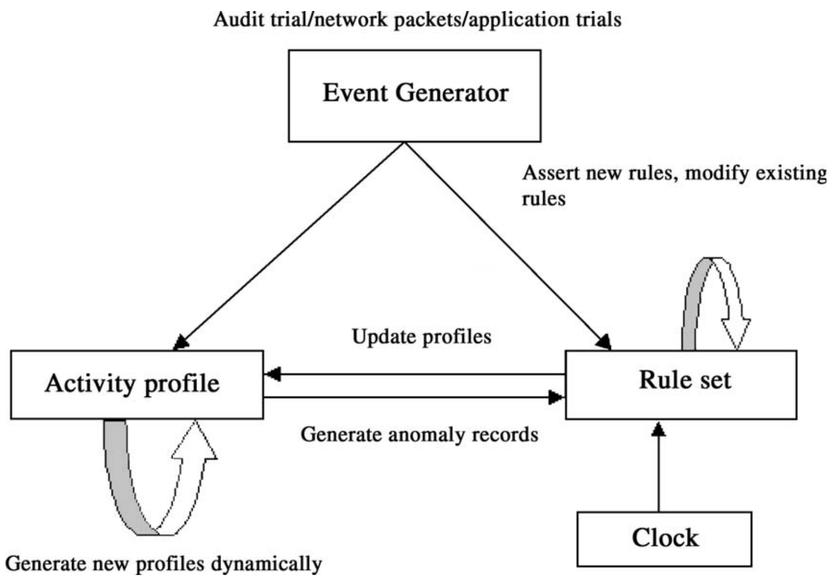
Audit trial/network packets/application trials
Event Generator
Assert new rules, modify existing rules
Update profiles
Activity profile
Rule set
Generate anomaly records
Clock
Generate new profiles dynamically

Fig. 1. A generic intrusion detection model (Kumar, 1995).

behavior with respect to a set of intrusion detection measures. Profiles are updated periodically, thus allowing the system to learn new behavior as users alter their behavior. These profiles are used to compare the user behavior and informing significant deviation from them as the intrusion. IDES also uses the expert system concept to detect misuse intrusions. The advantage of this approach is that it adaptively learns the behavior of users, which is thus potentially more sensitive than human experts. This system has several disadvantages. The system can be trained for certain behavior gradually making the abnormal behavior as normal, which may make the intruders undetected. Determining the threshold above which an intrusion should be detected is a difficult task. Setting the threshold too low results in false positives (normal behavior detected as an intrusion) and setting it too high results in false negatives (an intrusion undetected). Attacks, which occur by sequential dependencies, cannot be detected, as statistical analysis is insensitive to order of events (Lunt et al., 1992).

Predictive pattern generation uses a rule base of user profiles defined as statistically weighted event sequences (Teng and Chen, 1990). This method of intrusion detection attempts to predict future events based on events that have already occurred. This system develops sequential rules of the from

$$E1 - E2 - E3 \rightarrow (E4 = 94\%; E5 = 6\%),$$

where the various *E*'s are events derived from the security audit trail, and the percentage on the right-hand side of the rule represent the probability of occurrence of each of the consequent events given the occurrence of the antecedent sequence. This would mean that for the sequence of observed events $E1$ followed by $E2$ followed by $E3$, the probability of event $E4$ occurring is 94% and that of $E5$ is 6%. The rules are generated inductively with an information theoretic algorithm that measures the applicability of rules in terms of coverage and predictive power. An intrusion is detected if the observed sequence of events matches the left-hand side of the rule but the following events significantly deviate from the right-hand side of the rule. The main advantages of this approach include its ability to detect and respond quickly to anomalous behavior, easier to detect users who try to train the system during its learning period. The main problem with the system is its inability to detect some intrusions if that particular sequence of events have not been recognized and created into the rules.

The State transition analysis approach uses the state transitions of the system to identify intrusions. This method constructs the state transition diagram, which is the graphical representation of intrusion behavior as a series of state changes that lead from an initial secure state to a target compromised state. State transition diagrams list only the critical events that must occur for the successful completion of the intrusion. Using the audit trail as input, an analysis tool can be developed to compare the state changes produced by the user to state transition diagrams of known penetrations. State transition diagrams are written to correspond to the states of an actual computer system, and these diagrams form the basis of a rule-based expert system for detecting penetrations, called the state transition analysis tool (STAT) (Porras, 1992). The STAT prototype is implemented in unix state transition analysis tool (USTAT) (Ilgun, 1992) on UNIX-based systems. The main advantage of the method is that it detects intrusions independent of the audit trial record. It is also able to detect cooperative attacks, variations to known attacks and attacks spanned across multiple user sessions. This system has to be used along with some anomaly detector, because USTAT can detect only misuse intrusions. The disadvantages of the system are that it can only construct patterns from sequences of events but not from more complex forms and therefore some attacks cannot be detected, as they cannot be modeled with state transitions.

The Keystroke monitoring technique utilizes a user's keystrokes to determine the intrusion attempt. The main approach is to pattern match the sequence of keystrokes to some predefined sequences to detect the intrusion. The main problems with this approach is a lack of support from the operating system to capture the keystroke sequences. Furthermore, there are also many ways of expressing the sequence of keystrokes for the same attack. Some shell programs like *bash*, *ksh* have the user definable aliases utility. These aliases make it difficult to detect the intrusion attempts using this technique unless some semantic analysis of the commands is used. Automated attacks by malicious executables cannot be detected by this technique as they only analyze keystrokes.

In an expert system, knowledge about a problem domain is represented by a set of rules. These rules consist of two parts, antecedent, which defines when the rule should be applied and consequent, which defines the action(s) that should be taken if its antecedent is satisfied. A rule is fired when pattern-matching techniques determine that observed data matches or satisfies the antecedent of a rule. The rules may recognize single auditable events that represent significant danger to the system by themselves, or they may recognize a sequence of events that represent an entire penetration scenario. There are some disadvantages with the expert system method. An intrusion scenario that does not trigger a rule will not be detected by the rule-based approach. Maintaining and updating a complex rule-based system can be difficult. Since the rules in the expert system have to be formulated by a security professional, the system performance would depend on the quality of the rules.

The model-based approach attempts to model intrusions at a higher level of abstraction than audit trail records. The objective is to build scenario models that represent the characteristic behavior of intrusions. This allows administrators to generate their representation of the penetration abstractly, which shifts the burden of determining what audit records are part of a suspect sequence to the expert system. This technique differs from current rule-based expert system techniques, which simply attempt to pattern match audit records to expert rules. The model-based approach of (Garvey and Lunt, 1991) consists of three parts, namely, anticipator, planner and interpreter. The anticipator generates the next set of behaviors to be verified in the audit trail based on the current active models and passes these sets to the planner. The planner determines how the hypothesized behavior is reflected in the audit data and translates it into a system-dependent audit trail match. The interpreter then searches for this data in the audit trail. The system collects the information in this manner until a threshold is reached, and then it signals an intrusion attempt. Some of the drawbacks are that the intrusion patterns must always occur in the behavior it is looking for and patterns for intrusion must always be distinguishable from normal behavior and also easily recognizable.

The pattern matching (Kumar, 1995) approach encodes known intrusion signatures as patterns that are then matched against the audit data. Intrusion signatures are classified using structural inter relationships among the elements of the signatures. These structural interrelationships are defined over high level events or activities, which are themselves, defined in terms of low-level audit trail events. This categorization of intrusion signatures is independent of any underlying computational framework of matching. The patterned signatures are matched against the audit trails and any matched pattern can be detected as intrusion. Intrusions can be understood and characterized in terms of the structure of events needed to detect them. Model of pattern matching is implemented using colored petrinets in IDIOT (Kumar and Spafford, 1994). This system has several advantages. The system can be clearly separated into three parts, intrusion signatures as patterns, the audit trails as an abstracted event stream and the detector as a pattern matcher. This makes different solutions to be substituted for each component without changing the overall structure of the system. Pattern specifications are declarative, which means pattern representation of intrusion signatures can be specified by defining what needs to be matched than how it is matched. Declarative specification of patterns enables them to be exchanged across different operating systems with different audit trails. Intrusion signatures can be moved across sites without rewriting them as the representation of patterns is standardized. However, there are few problems in this approach. Constructing patterns from attack scenarios is a difficult problem and needs human expertise. Attack scenarios that are known and constructed into patterns by the system can only be detected. Attacks involving spoofing and passive methods of attack like wire-tapping cannot be detected.

The data mining approach to intrusion detection was first implemented in mining audit data for automated models for intrusion detection (MADAMID) (Lee et al., 1999). The data mining process of building intrusion detection models is depicted in

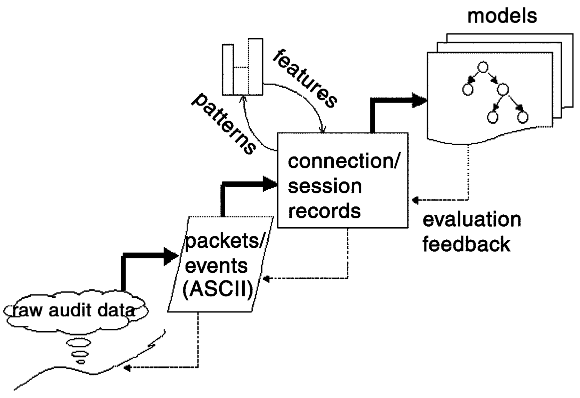

Fig. 2. Data mining process of building Intrusion detection models (Lee, 1999).

Fig. 2 (Lee and Stolfo, 1998). Raw data is first converted into ASCII network packet information, which in turn is converted into connection level information. These connection level records contain within connection features like service, duration etc. Data mining algorithms are applied to this data to create models to detect intrusions (Grossman et al., 1998). Data mining algorithms used in this approach include rule-based classification algorithm (RIPPER), meta-classifier, frequent episode algorithm and association rules. These algorithms are applied to audit data to compute models that accurately capture the actual behavior of intrusions as well as normal activities. The main advantage of this system is automation of data analysis through data mining, which enables it to learn rules inductively replacing manual encoding of intrusion patterns. The problem is it deals mainly with misuse detection, hence some novel attacks may not be detected. Audit data analysis and mining (ADAM) (Barbara et al., 2001) also uses data mining methods. Combination of association rules and classification algorithm were used to discover attacks in audit data. Association rules are used to gather necessary knowledge about the nature of the audit data as the information about patterns within individual records can improve the classification efficiency. This system has two phases, training phase and detection phase. In the training phase, a database of frequent item sets is created for the attack-free items using only the attack-free data set. This serves as a profile against which frequent item sets found later will be compared. Next a sliding-window, on-line algorithm is used to find frequent item sets in the last D connections and compares them with those stored in the attack-free database, discarding those that are deemed normal. In this phase, the classifier is also trained to detect the attack. In the detection phase, a dynamic algorithm is used to produce item sets that are considered as suspicious and used by the classification algorithm already learned to classify the item set as attack, false alarm (normal event) or as unknown. Unknown attacks are the ones which are not able to be detected either as false alarms or as known attacks. This method also uses the pseudo-Bayes estimator to classify the attack to avoid the dependency on the data. This method attempts to detect only anomaly attacks.

Artificial neural network (ANN) is another data mining approach taken in Intrusion detection (Fox et al., 1990). An ANN consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. Neural networks have been used both in anomaly intrusion detection as well as in misuse intrusion detection. In Debar et al. (1992), the system learns to predict the next command based on a sequence of previous commands input by a user. Here a shifting window of w recent commands is used. The predicted command of the user is compared with the actual command of the user and any deviation is signaled as intrusion. The window size w places an important role, because if w is too small, there will be many false positives and if it is too big some attacks may not be detected. Neural network intrusion detector (NNID) (Ryan et al., 1998) identifies intrusions based on the distribution of commands used by the user. This system has three phases. In the first phase it collects training data from the audit logs for each user for some period and constructs a vector from the collected data to represent the command execution distribution for each user. In the second phase, the neural network is trained to identify the user based on these command distribution vectors. In the final phase the network identifies the user for each new command distribution vector. If the networks identify a user as being different from the actual user, an anomaly intrusion is signalled. A neural network for misuse detection is implemented in two ways (Cannady, 1998). The first approach incorporates the neural network component into the existing or modified expert system. This method uses the neural network to filter the incoming data for suspicious events and forwards them to the expert system. This improves the effectiveness of the detection system. The second approach uses the neural network as a stand alone misuse detection system. In this method, the neural network receivse data from the network stream and analyzes it for misuse intrusion. There are several advantages to this approach. It has the ability to learn the characteristics of misuse attacks and identify instances that are unlike any which have been observed before by the network. It has a high degree of accuracy to recognize known suspicious events. Neural networks work well on noisy data. The inherent speed of neural networks is very important for real time intrusion detection. The main problem is in the training of neural networks, which is important for obtaining efficient neural networks. The training phase also requires a very large amount of data.

SVM are learning machines that plot the training vectors in high-dimensional feature space, labeling each vector by its class. SVMs classify data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in the feature space. SVM have proven to be a good candidate for intrusion detection because of their speed. SVM are scalable as they are relatively insensitive to the number of data points. Therefore the classification complexity does not depend on the dimensionality of the feature space; hence, they can potentially learn a larger set of patterns and scale better than neural networks (Mukkamala et al., 2003).

Neuro-fuzzy (NF) computing combines fuzzy inference with neural networks (Abraham, 2001). Knowledge expressed in the form of linguistic rules can be used to build an FIS, With data, ANNs can be built. For building an FIS, the user has to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches are complementary and therefore it is natural to consider building an integrated system combining these two concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base is an advantage from the viewpoint of ANN. An Adaptive neuro-fuzzy IDS is proposed in Shah et al. (2004).

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 hide in high-dimensional data. An IDS based on MARS technology is proposed in Mukkamala et al. (2004b).

LGP is a variant of the conventional genetic programming (GP) technique that acts on linear genomes. Its main characteristics in comparison to tree-based GP lies in the fact that computer programs are evolved 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. It always has to be represented as a piece of machine code finally, as fitness evaluation requires physical execution of the individuals. An LGP-based IDS is presented in Mukkamala et al. (2004a).

Intrusion detection systems based on the human immunological system have been proposed in Esponda et al. (2004), Hofmeyr and Forrest (1999). Forrest et al. (Esponda et al., 2004) propose a formal framework for anomaly detection in computer systems, inspired by the characteristics of the natural immune system. Hofmeyr and Forrest (1999) applied the concepts derived from natural immune system to design and test an artificial immune system to detect network intrusion. They specifically mentioned 4 important characteristics of natural immune system that they think define immunity. They are diversity, distributed nature, error tolerance and dynamic nature. They designed the detector analogous to the T and B-lymphocytes that are found in the human immunological system.

3. Hybrid approaches for intrusion detection

3.1. Support vector machines (SVM)

An SVM maps input (real-valued) feature vectors into a higher-dimensional feature space through some nonlinear mapping. SVMs are developed on the principle of structural risk minimization (Vapnik, 1995). Structural risk minimization seeks to find a hypothesis h for which one can find lowest probability of error whereas the traditional learning techniques for pattern recognition are based on the minimization of the empirical risk, which attempt to optimize the performance of the learning set. Computing the hyper plane to separate the data points i.e. training an SVM leads to a quadratic optimization problem. SVM uses a linear separating hyper plane to create a classifier but all the problems cannot be separated linearly in the original input space. SVM uses a feature called kernel to solve this problem. The Kernel transforms linear algorithms into nonlinear ones via a map into feature spaces. There are many kernel functions; including polynomial, radial basis functions, two layer sigmoid neural nets etc. The user may provide one of these functions at the time of training the classifier, which selects support vectors along the surface of this function. SVMs classify data by using these support vectors, which are members of the set of training inputs that outline a hyper plane in feature space (Joachims, 1998).

3.2. Decision trees (DT)

DT induction is one of the classification algorithms in data mining. The classification algorithm is inductively learned to construct a model from the pre-classified data set (Brieman et al., 1984). Inductive learning means making general assumptions from the specific examples in order to use those assumptions to classify unseen data. The inductively learned model of classification algorithm is known as classifier. Classifier may be viewed as mapping from a set of attributes to a particular class. Data items are defined by the values of their attributes and X is the vector of their values $\{x_1, x_2, \ldots, x_n\}$, where the value is either numeric or nominal. Attribute space is defined as the set containing all possible attribute vectors and is denoted by Z. Thus X is an element of Z ($X \in Z$). The set of all classes is denoted by $C = \{c_1, c_2, \ldots, c_n\}$. A classifier assigns a class $c \in C$ to every attribute of the vector $X \in Z$. The classifier can be considered as a mapping f, where f: $X \rightarrow C$. This classifier is used to classify the unseen data with a class label. A DT classifies the given data item using the values of its attributes. The DT is initially constructed from a set of pre-classified data. Each data item is defined by values of the attributes. The main issue is to select the attributes which best divides the data items into their classes. According to the values of these attributes the data items are partitioned. This process is recursively applied to each partitioned subset of the data items. The process terminates when all the data items in the current subset belongs to the same class. A DT consists of nodes, leaves and edges. A node of a DT specifies an attribute by which the data is to be partitioned. Each node has a number of edges, which are labeled according to a possible value of edges and a possible value of the attribute in the parent node. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data. Induction of the DT uses the training data, which is described in terms of the attributes. The main problem here is deciding the attribute, which will best partition the data into various classes. The ID3 algorithm uses the information theoretic approach to solve this problem. Information theory uses the concept entropy, which measures the impurity of data items. Entropy specifies the number of bits required to encode the classification of a data item. The value of entropy is small when the class distribution is uneven, that is when all the data items belong to one class. The entropy value is higher when the class distribution is more even, that is when the data items have more classes. Information gain is a measure on the utility of each attribute in classifying the data items. It is measured using the entropy value. Information gain measures the decrease of the weighted average impurity (entropy) of the attributes compared with the impurity of the complete set of data items. Therefore, the attributes with the largest information gain are considered as the most useful for classifying the data items. To classify an unknown object, one starts at the root of the DT and follows the branch indicated by the outcome of each test until a leaf node is reached. The name of the class at the leaf node is the resulting classification. DT induction has been implemented with several algorithms. Some of them are ID3 (Quinlan, 1986) and later on it was developed into C4.5 (Quinlan, 1993) and C5.0. C4.5 is an extension of the basic ID3 algorithm. C4.5 handles continuous attributes and is able to choose an appropriate attribute selection measure. It also deals with missing attribute values and improves computation efficiency. C4.5 builds the tree from a set of data items using the best attribute to test in order to divide the data item into subsets and then it uses the same procedure on each subset recursively. The best attribute to divide the subset at each stage is selected using the information gain of the attributes. For nominal valued attributes, a branch for each value of the attribute is formed, whereas for numeric valued attributes, a threshold is found, thus forming two branches.

3.3. Hybrid decision tree–SVM (DT–SVM) approach

A hybrid intelligent system uses the approach of integrating different learning or decision-making models. Each learning model works in a different manner and exploits different set of features. Integrating different learning models gives better performance than the individual learning or decision-making models by reducing their individual limitations and exploiting their different mechanisms. In a hierarchical hybrid intelligent system each layer provides some new information to the higher level (Abraham, 2002). The overall functioning of the system depends on the correct functionality of all the layers. Fig. 3 shows the architecture of the hybrid intelligent system with DT and SVM.

The data set is first passed through the DT and node information is generated. Node information is determined according to the rules generated by the DT. Terminal nodes are numbered left to right starting with 1 as shown in the Fig. 4. All the data set records are assigned to one of the terminal nodes, which represent the particular class or subset. The terminal nodes show either red or blue color representing either normal or attack. This node information (as an additional attribute) along with the original set of attributes is passed through the SVM to obtain the final output. The key idea here is to investigate whether the node information provided by the DT will improve the performance of the SVM.

3.4. Ensemble approach

Empirical observations show that different classifiers provide complementary information about the patterns to be classified. Although for a particular problem

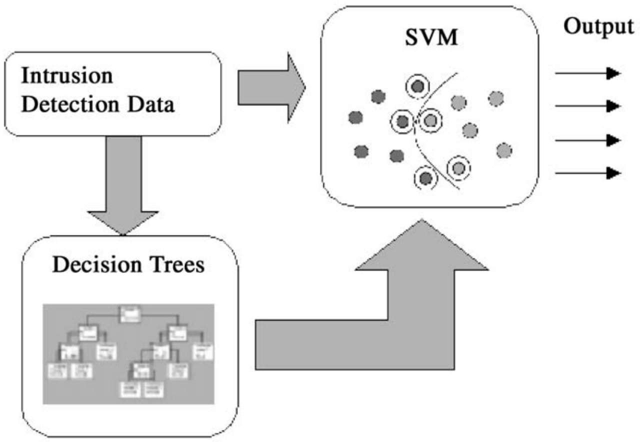

Fig. 3. Hybrid decision tree–SVM model.

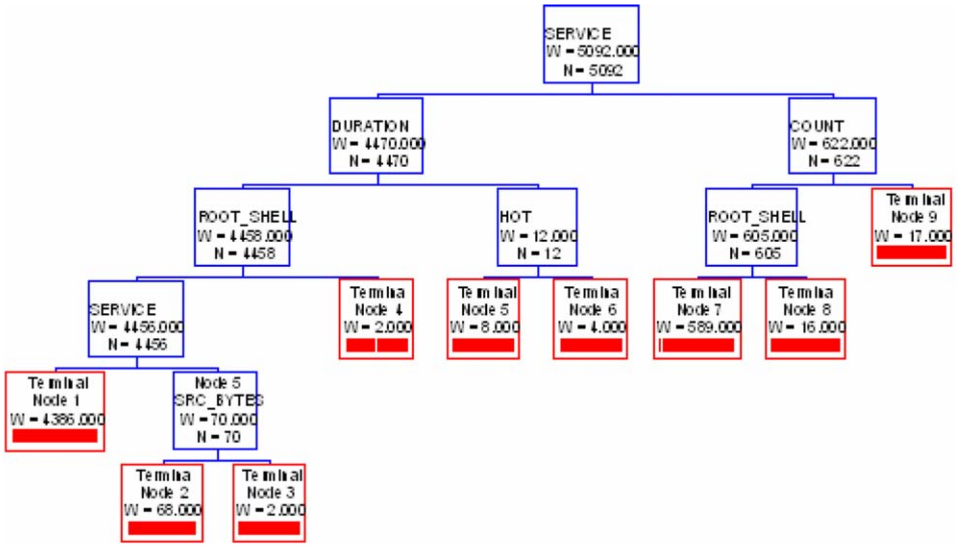

Fig. 4. Decision tree output structure.

one classifier works better than the other, a set of misclassified patterns would not necessarily overlap. This different information combined together yields better performance than individual classifiers. The idea is not to rely on a single classifier for decision on an intrusion; instead information from different individual classifiers is combined to take the final decision, which is popularly known as the ensemble approach. The effectiveness of the ensemble approach depends on the accuracy and diversity of the base classifiers.

We used the highest scored class as the final output among the base classifier outputs (DT, SVM and DT–SVM). According to the performance on the training data each classifier is assigned different weights. Using these weights and the outputs of the classifiers, scores were calculated. For example, for class 1 if the DT works best, followed by the DT–SVM and SVM model, the DT is assigned the highest weight, followed by the hybrid DT–SVM model and SVM is assigned the lowest weight. For five different classes each classifier has different weights depending on their performance on the training data. So for a particular data record if all of them have different opinions, their scores are considered and the highest score is declared as the actual output of the ensemble approach. The architecture of the ensemble approach is depicted in Fig. 5.

4. Experiment setup and performance evaluation

The KDD Cup 1999 Intrusion detection contest data (KDD cup 99 Intrusion detection data set) is used in our experiments. This data was prepared by the 1998

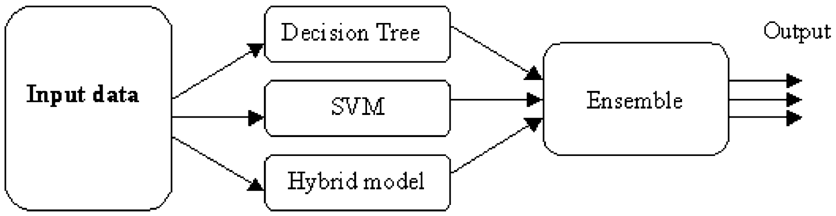

Fig. 5. Architecture of ensemble approach.

DARPA Intrusion Detection Evaluation program by MIT Lincoln Labs (MIT Lincoln Laboratory). Lincoln labs acquired nine weeks of raw TCP dump data. The raw data was processed into connection records, which consist of about 5 million connection records. The data set contains 24 attack types. These attacks fall into four main categories:

1. *Denial of service (DOS)*: In this type of attack an attacker makes some computing or memory resources too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Apache2, Back, Land, Mailbomb, SYN Flood, Ping of death, Process table, Smurf, Teardrop.
2. *Remote to user (R2L)*: In this type of attack an attacker who does not have an account on a remote machine sends packets to that machine over a network and exploits some vulnerability to gain local access as a user of that machine. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock.
3. *User to root (U2R)*: In this type of attacks an attacker starts out with access to a normal user account on the system and is able to exploit system vulnerabilities to gain root access to the system. Examples are Eject, Loadmodule, Ps, Xterm, Perl, Fdformat.
4. *Probing*: In this type of attacks an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are Ipsweep, Mscan, Saint, Satan, Nmap.

The data set has 41 attributes for each connection record plus one class label. R2L and U2R attacks don't 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. Therefore, 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 two phases, namely, a training and a testing phase. In the training phase the system constructs a model using the training data to give maximum generalization accuracy (accuracy on unseen data). The test data is passed through the constructed model to detect the intrusion in the testing phase. Besides the four different types of attacks mentioned above we also have to detect the normal class. The data set for our experiments contained 11982 records (KDD cup 99 Intrusion detection data set), which were randomly generated from the MIT data set. Random generation of data include the number of data from each class proportional to its size, except that the smallest class is completely included. This data set is again divided into training data with 5092 records and testing data with 6890 records. All the intrusion detection models are trained and tested with the same set of data. As the data set has five different classes we perform a 5-class classification. The normal data belongs to *class1*, probe belongs to *class*2, denial of service (DoS) belongs to *class*3, user to root (U2R) belongs to *class*4 and remote to local (R2L) belongs to *class*5. Experiments were performed using an AMD Athlon, 1.67 GHz processor with 992 MB of RAM.

4.1. Decision tree

Although DT are capable of handling a 5-class classification problem, we used a binary DT classifier in this work so that comparisons with the SVM classifier which is a binary classifier, would make sense. We constructed five different classifiers. The data is partitioned into the two classes of "Normal" and "Attack" patterns where Attack is the collection of four classes (Probe, DOS, U2R, and R2L) of attacks. The objective is to separate normal and attack patterns. We repeat this process for all the five classes. First a classifier was constructed using the training data and then testing data was tested with the constructed classifier to classify the data into normal or attack. Table 1 summarizes the results of the test data. It shows the training and testing times of the classifier in seconds for each of the five classes and their accuracy.

4.2. Support vector machines

The Kernel option defines the feature space in which the training set examples will be classified. Our trial and error experiments and a previous study (Ali and Abraham, 2002) showed that a polynomial kernel option often performs well on most data sets. We therefore used the polynomial kernel for our experiments. From our experiments, we observed that for different classes of data, different polynomial degrees gave different performance and the empirical results using test data set are presented in Table 2.

Table 1
Performance of decision trees

Attack type	Training time (s)	Testing time (s)	Accuracy (%)
Normal	1.53	0.03	99.64
Probe	3.09	0.02	99.86
DOS	1.92	0.03	96.83
U2R	1.16	0.03	68.00
R2L	2.36	0.03	84.19

Table 2
Classification accuracy for different polynomial kernel degrees

Attack type	Polynomial degree		
	1	2	3
Normal	99.64	99.64	99.64
Probe	98.57	64.85	61.72
DOS	70.99	99.92	99.78
U2R	40.00	40.00	40.00
R2L	33.92	31.44	28.06

Table 3
Performance of the SVM

Attack type	Training time (s)	Testing time (s)	Accuracy (%)
Normal	5.02	0.13	99.64
Probe	1.33	0.13	98.57
DOS	19.24	2.11	99.92
U2R	3.05	0.95	40.00
R2L	2.02	0.13	33.92

As SVMs handle binary class classification problems, we employed five SVMs for detecting the five types of attacks. The classifier learns from the training data and is used on the test data to classify the data into normal or attack patterns. This process is repeated for all classes. The results are summarized in Table 3.

Our experiments show that the DT gives better accuracy for Probe, R2L and U2R classes compared to SVM and it gives the worst accuracy for detecting DoS class of attacks. For Normal class both methods give the same performance. There is only a small difference in the accuracy for Normal, Probe and DOS classes for DT and SVM but there is a significant difference for U2R and R2L classes. Since these two classes have small training data compared to other classes it seems that DT gives good accuracy with small training data sets. The training time and testing times are also less for the DT compared to the SVM.

4.3. *Hybrid decision tree–SVM*

A hybrid DT–SVM model has two steps for constructing the classifier. The data sets were first passed through the DT and the node information was generated. Training and test data along with the node information is given to the SVM. SVM gives the final output of the hybrid DT–SVM.

The performance of the hybrid DT–SVM is illustrated in Table 4 and 5. Hybrid DT–SVM works better than the individual DT and SVM for normal class. For Probe, U2R and R2L classes it performed better than an individual SVM approach.

Table 4
Performance comparisons of three classifiers

Attack type	Decision tree accuracy (%)	SVM accuracy (%)	Hybrid decision tree—SVM accuracy (%)
Normal	99.64	99.64	99.70
Probe	99.86	98.57	98.57
DOS	96.83	99.92	99.92
U2R	68.00	40.00	48.00
R2L	84.19	33.92	37.80

Table 5
Performance of ensemble approach

Attack type	Accuracy (%)			
	Decision trees	SVM	Hybrid decision tree–SVM	Ensemble approach
Normal	99.64	99.64	99.70	99.70
Probe	99.86	98.57	98.57	100.00
DOS	96.83	99.92	99.92	99.92
U2R	68.00	40.00	48.00	68.00
R2L	84.19	33.92	37.80	97.16

From the above results we can conclude that although the node information generated by the DT did enhance the performance of SVM, on the whole the hybrid DT–SVM model did not give the expected performance.

4.4. Ensemble approach

In this approach, we first construct DT, SVM and hybrid DT–SVM classifiers individually to obtain a good generalization performance (optimizing the model for performance on unseen data rather than the training data). Test data is passed through each individual model and the corresponding outputs are used to decide the final output. The performance of the ensemble approach is presented in Table 5. Empirical results depict that the proposed ensemble approach gives better performance for detecting probes and U2R attacks than all the three individual models.

The Ensemble approach classifies most of them correctly by picking up all the classes, which are correctly classified by all the three classifiers. As expected the ensemble approach exploits the differences in misclassification and improves the overall performance. As evident from Table 5, all the classifiers considered so far could not perform well for detecting all the attacks. To take advantage of the performance of the different classifiers a hierarchical hybrid intelligent system is proposed as depicted in Fig. 6. The hybrid IDS model makes uses of individual,

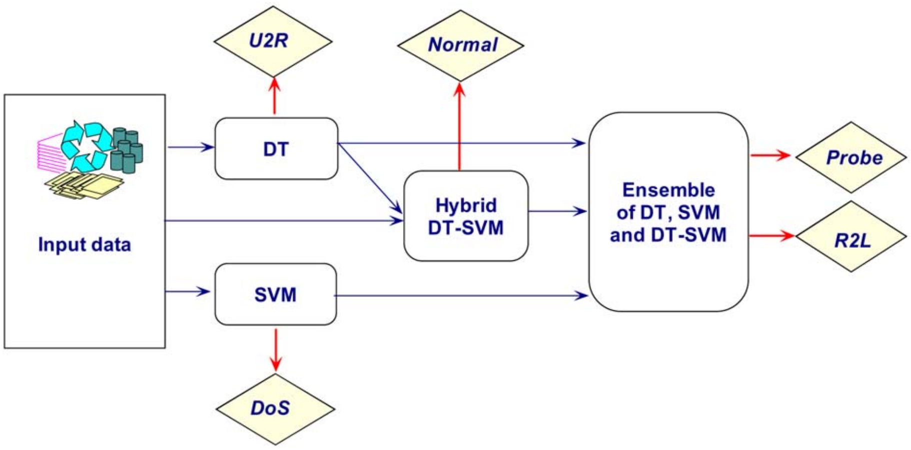

Fig. 6. IDS based on a hierarchical intelligent system.

hybrid and ensemble approaches to maximize the computational efficiency and detection accuracy for each class. The proposed model (Fig. 6) would therefore give the overall best performance accuracy for the individual attacks as depicted in Table 5.

5. Conclusions

In this research, we have investigated some new techniques for intrusion detection and evaluated their performance based on the benchmark KDD Cup 99 Intrusion data. We have explored DT and SVM as intrusion detection models. Next we designed a hybrid DT–SVM model and an ensemble approach with DT, SVM and DT–SVM models as base classifiers. Empirical results reveal that DT gives better or equal accuracy for Normal, Probe, U2R and R2L classes. The hybrid DT–SVM approach improves or delivers equal performance for all the classes when compared to a direct SVM approach.

The Ensemble approach gave the best performance for Probe and R2L classes. The ensemble approach gave 100% accuracy for Probe class, and this suggests that if proper base classifiers are chosen 100% accuracy might be possible for other classes too. Finally, we propose a hierarchical intelligent IDS model to make optimum use of the best performances delivered by the individual base classifiers and the ensemble approach.

References

Abraham A. In: Jose M., Alberto P., editors, Neuro-fuzzy systems: state-of-the-art modeling techniques, connectionist models of neurons, learning processes, and artificial intelligence, Lecture Notes in Computer Science, vol. 2084. Germany, Granada, Spain: Springer; 2001. p. 269–76.

Abraham A. Intelligent systems: architectures and perspectives, recent advances in intelligent paradigms and applications. In: Abraham A, Jain L, Kacprzyk J, editors. Studies in fuzziness and soft computing. Germany: Springer; 2002. p. 1–35 [Chapter 1].

Ali ABMS, Abraham A. An empirical comparison of kernel selection for support vector machines. In: Proceedings of the second international conference on hybrid intelligent systems: design, management and applications. The Netherlands: IOS Press; 2002. p. 321–30.

Anderson JP. Computer security threat monitoring and surveillance. Technical Report, Fort Washington, Pennsylvania: James P Anderson Co.; April 1980.

Barbara D, Couto J, Jajodia S, Wu N. ADAM: a testbed for exploring the use of data mining in intrusion detection. SIGMOD Record 2001;30(4):15–24.

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

Cannady J. Artificial neural networks for misuse detection. National Information Systems Security Conference 1998.

Debar H, Becke M, Siboni D. A neural network component for an intrusion detection system. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy 1992.

Denning DE. An intrusion detection model. IEEE Transactions on Software Engineering 1997:222–8.

Esponda F, Forrest S, Helman P. A formal framework for positive and negative detection. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 2004;34(1).

Fox KL, Henning RR, Reed JH, Simonian R. A neural network approach towards Intrusion detection. In: Proceedings of the 13th national computer security conference, Washington, DC, October 1990. p. 125–34.

Garvey TD, Lunt TF. Model based intrusion detection. In: Proceedings of the 14th national computer security conference, October 1991. p. 372–85.

Grossman R, Kasif S, Moore R, Rocke D, Ullman J. Data Mining Research: Opportunities and Challenges, A Report of Three NSF Workshops on Mining Large, Massive, and Distributed Data, January 1998.

Heady R, Luger G, Maccabe A, Servilla M. The architecture of a network level intrusion detection system. Technical Report, Department of Computer Science, University of New Mexico, August 1990.

Hofmeyr SA, Forrest S. Immunity by design: an artificial immune system. In: Proceedings GECCO Conference, 1999.

Ilgun K. USTAT: A real-time intrusion detection system for UNIX. Master thesis, University of California, Santa Barbara, November 1992.

Joachims T. Making large-scale SVM learning practical. LS8-Report, University of Dortmund, LS VIII-Report, 1998.

KDD cup 99 Intrusion detection data set. 〈http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz〉

Kumar S. Classification and detection of computer intrusions. PhD thesis, Department of Computer Science, Purdue University, August 1995.

Kumar S, Spafford EH. An application of pattern matching in intrusion detection. Technical Report CSD-TR-94-013, Purdue University, 1994.

Kumar S, Spafford EH. A software architecture to support misuse intrusion detection. In: Proceedings of the 18th national information security conference, 1995. p. 194–204.

Lee W. A data mining framework for constructing features and models for intrusion detection systems. PhD thesis, Computer Science Department, Columbia University, June 1999.

Lee W, Stolfo S. Data mining approaches for intrusion detection. In: Proceedings of the 7th USENIX security symposium, 1998.

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 1999.

Lunt T. Detecting intruders in computer systems. In: Proceedings of the 1993 conference on auditing and computer technology 1993.

Lunt T, Tamaru A, Gilham F, Jagannathan R, Neumann P, Javitz H. A real-time intrusion detection expert system (IDES)—final technical report. Technical Report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

MIT Lincoln Laboratory. ⟨http://www.ll.mit.edu/IST/ideval/⟩

Mukkamala S, Sung AH, Abraham A. Intrusion detection using ensemble of soft computing paradigms, third international conference on intelligent systems design and applications, intelligent systems design and applications, advances in soft computing. Germany: Springer; 2003. p. 239–48.

Mukkamala S, Sung AH, Abraham A. Modeling intrusion detection systems using linear genetic programming approach, The 17th international conference on industrial & engineering applications of artificial intelligence and expert systems, innovations in applied artificial intelligence. In: Robert O., Chunsheng Y., Moonis A., editors. Lecture Notes in Computer Science, vol. 3029. Germany: Springer; 2004a. p. 633–42.

Mukkamala S, Sung AH, Abraham A, Ramos V. Intrusion detection systems using adaptive regression splines. In: Seruca I, Filipe J, Hammoudi S, Cordeiro J, editors. Proceedings of the 6th international conference on enterprise information systems, ICEIS'04, vol. 3, Portugal. 2004b. p. 26–33 [ISBN:972-8865-00-7].

Porras PA. STAT: a state transition analysis tool for intrusion detection. Master's thesis, Computer Science Department, University of California, Santa Barbara, July 1992.

Quinlan JR. Induction of decision trees. Machine Learning 1986;1:81–106.

Quinlan JR. C4.5: programs for machine learning. Log Altos, CA: Morgan Kaufmann; 1993.

Ryan J, Lin MJ, Miikkulainen R. Intrusion detection with neural networks. advances in neural information processing systems, vol. 10. Cambridge, MA: MIT Press; 1998.

Shah K, Dave N, Chavan S, Mukherjee S, Abraham A, Sanyal S. Adaptive neuro-fuzzy intrusion detection system. IEEE International Conference on Information Technology: Coding and Computing (ITCC'04), vol. 1. USA: IEEE Computer Society; 2004. p. 70–4.

Summers RC. Secure computing: threats and safeguards. New York: McGraw-Hill; 1997.

Sundaram A. An introduction to intrusion detection. ACM Cross Roads 1996;2(4).

Teng HS, Chen K, Lu SC. Security audit trail analysis using inductively generated predictive rules. In: Proceedings of the 11th national conference on artificial intelligence applications. Piscataway, NJ: IEEE, IEEE Service Center; March 1990. p. 24–9.

Vapnik VN. The nature of statistical learning theory. Berlin: Springer; 1995.