DISTRIBUTED INTRUSION DETECTION SYSTEMS: A COMPUTATIONAL INTELLIGENCE APPROACH

Ajith Abraham and Johnson Thomas*

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

*Department of Computer Science, Oklahoma State University, USA
Email: jpt@okstate.edu

Abstract.

Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability. An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. 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). A Distributed IDS (DIDS) consists of several IDS over a large network (s), all of which communicate with each other, or with a central server that facilitates advanced network monitoring. In a distributed environment, DIDS are implemented using co-operative intelligent agents distributed across the network(s). This chapter presents a framework for a distributed intrusion detection system comprising of a multi-agent framework with computational intelligent techniques to reduce the data features to create lightweight detection systems and a hybrid intelligent system approach to improve the detection accuracy.

1. Introduction

The traditional prevention techniques such as user authentication, data encryption, avoiding programming errors and firewalls are used as the first line of defense for computer security. If a password is weak and is compromised, user authentication cannot prevent unauthorized use, firewalls are vulnerable to errors in configuration and ambiguous or undefined security policies. They are generally unable to protect against malicious mobile code, insider attacks and unsecured modems. Programming errors cannot be avoided as the complexity of the system and

application software is changing rapidly leaving behind some exploitable weaknesses. Intrusion detection therefore becomes a necessity as an additional wall for protecting systems despite the prevention techniques. Recently, Intrusion Detection Systems (IDS) have been used in monitoring attempts to break security, which provides important information for timely countermeasures. 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 the system and application software to identify the intrusions. These patterns are encoded in advance and used to match against the user behavior to detect intrusion. Anomaly intrusion detection 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. In Distributed IDS (DIDS) conventional intrusion detection system are embedded inside intelligent agents and are deployed over a large network. In a distributed environment IDS agents communicate with each other, or with a central server. Distributed monitoring allows early detection of planned and coordinated attacks and thereby allowing the network administrators to take preventive measures. DIDS also helps to control the spreading of worms, improves network monitoring and incident analysis, attack tracing and so on. It also helps to detect new threats from unauthorized users, back-door attackers and hackers to the network across multiple locations, which are geographically separated. In a DIDS it is important to ensure that the individual IDS are light-weight and accurate.

Intrusion detection is a critical component of homeland security. Two important agencies that are part of the Office of Homeland Security are the Critical Infrastructure Assurance Office (CIAO) and the National Infrastructure Protection Center (NIPC). Leslie Wiser of the NIPC made the following statement to House Committee on Government Affairs: "…intrusions into critical systems are inevitable for the foreseeable future. Thus detection of these viruses, worms, and other intrusions is crucial if the U.S. Government and critical infrastructure owners and operators are going to be able to respond effectively " (Wiser, 2002). Clearly this applies to any nation facing the threat of terrorism. Information security and the protection of IT and communications infrastructure is a clear focus of Homeland security. Intrusion detection specifically, and intrusion management in general, will be important

components of that focus. Intrusion detection is fundamental to achieving secure operating environments in the future and is every bit as relevant to Homeland Security initiatives as it is to the commercial sector.

This chapter presents a computational intelligence approach for DIDS with a focus on improving the performance of the individual IDS agents by reducing the number of input features (light weight) and engaging hybrid intelligent systems for improving the detection accuracy.

2. Intrusion Detection Systems

2.1 Anomaly Detection

Anomaly detection assumes that an intrusion will always reflect some deviations from normal patterns. Anomaly detection can be divided into static and dynamic. A static anomaly detector is based on the assumption that there is a portion of system being monitored that should remain constant. 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. Static portions of the system can be represented as a binary bit string or a set of such strings (such as files). 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. Static anomaly detectors are meant for checking data integrity. Dynamic anomaly detectors require a definition of behavior to classify as normal or anomalous. Frequently, system designers employ the notion of event. Behavior is defined as a sequence of distinct actions that cause events that are recorded in audit records. Since audit records of operating system only record events of interest, then the only behavior that can be observed is that which results in an event in an audit record. Events may occur in a sequence. In some cases like distributed systems, partial ordering of events is sufficient. 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.

2.2 Misuse Detection

Misuse detection is based on the knowledge of system vulnerabilities and the 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 would be aware of all the known vulnerabilities and would

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 the intrusion scenarios to make sure that someone or combinations of people are not attempting to exploit known vulnerabilities. To perform this, each intrusion scenario must be described or modeled in some way. Generally, intrusion scenarios are quite specific. The main difference between the misuse techniques is in how they describe or model the bad behavior that constitutes an intrusion. Initial misuse detection systems used rules to describe the events indicative of intrusive actions that a security administrator looked for within the system. Large numbers of rules can be difficult to interpret if the *if-then* rules are not grouped by intrusion scenarios. This is because making modifications to the rule set can be difficult if the affected rules are spread out across the rule set. To overcome these difficulties, alternative intrusion scenario representations have been developed. These new rules impose organizational techniques that include model-based rule organization and state transition diagrams. Better rule organization allows the intrusion scenarios to be described in a more expressive and understandable way for the misuse detection system user. 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. Although most systems use audit records, they would be fundamentally the same if they were collecting live system information.

2.3. Related Research

Dorothy Denning (Denning, 1997) proposed the concept of intrusion detection as a solution to the problem of providing a sense of security in computer systems. The basic idea is that intrusion behavior involves abnormal usage of the system. Different techniques and approaches have been used in later developments. Some of the techniques used are statistical approaches, predictive pattern generation, expert systems, keystroke monitoring, state transition analysis, pattern matching, and data mining techniques. Figure 1 illustrates a simple network, which is protected using IDS.

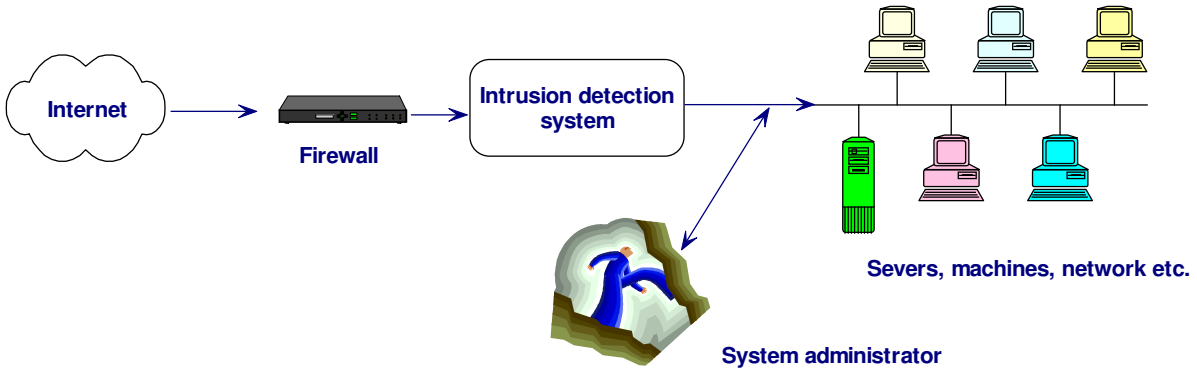

Figure 1. Network protection using conventional IDS

Statistical approaches compare the recent behavior of a user of a computer system with observed behavior and any significant deviation is considered as intrusion (Mukherjee, 1994). This approach requires construction of a model for normal user behavior. Any user behavior that deviates significantly from this normal behavior is flagged as an intrusion. Predictive pattern generation uses a rule base of user profiles defined as statistically weighted event sequences (Teng, 1990). This method of intrusion detection attempts to predict future events based on events that have already occurred. State transition analysis approach uses the state transitions of the system to identify the intrusions. State transition diagrams list only the critical events that must occur for the successful completion of the intrusion (Porras, 1992). Keystroke monitoring technique utilizes user's keystrokes to determine the intrusion attempt (Dunigan, 1999). 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 are lack of support from operating system to capture the keystroke sequences and also many ways of expressing the sequence of keystrokes for same attack. Expert systems have played an important role to build IDS (Ilgun, 1993). 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. The Model-based approach proposed by Garvey and Lunt (Garvey, 1991) 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 technique differs from the rule-based expert system technique, which simply attempt to pattern match audit records to expert rules. The Pattern matching (Kumar, 1995) approach encodes known intrusion signatures as patterns that are then matched against the audit data. A Model of pattern matching is implemented using colored Petri Nets in IDIOT (Kumar, 1994). Intrusion signature is represented with Petri Net, the start state and final state notion is used to define matching to detect the intrusion.

Data mining approaches for intrusion detection was first implemented in Mining Audit Data for Automated Models for Intrusion Detection (Lee, 1999). Raw data is converted into ASCII network packet information, which in turn is converted into connection level information. These connection level records contain within connection features like service, duration etc. Data mining algorithms are applied to this data to create models to detect intrusions. Neural networks have been used both in anomaly intrusion detection as well as in misuse intrusion detection. In the first approach of neural networks (Debar, 1992) for intrusion detection, the system learns to predict the next command based on a sequence of previous commands by a user. SVMs have proven to be a good candidate for intrusion detection because of its training speed and scalability. Besides SVMs are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of the feature space, so they can potentially learn a larger set of patterns and scale better than neural networks (Mukkamala, 2003).

Neuro-fuzzy computing is a popular framework for solving complex problems. An Adaptive neuro-fuzzy IDS is proposed in (Shah, 2004). Multivariate Adaptive Regression Splines (MARS) is an innovative approach that automates the building of accurate predictive models for continuous and binary dependent variables (Friedman, 1991). It excels at finding optimal variable transformations and interactions, and the complex data structure that often hides in high-dimensional data. An IDS based on MARS technology is proposed in (Mukkamala, 2004b). Linear Genetic Programming (LGP) is a variant of the conventional Genetic Programming (GP) technique that acts on linear genomes. Its main characteristics in comparison to tree-based GP lies in 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, finally it always has to be represented as a piece of machine code, as fitness evaluation requires physical execution of the individuals. LGP based IDS is presented in (Mukkamala, 2004a).

3. Distributed Intrusion Detection System (DIDS)

A number of IDSs have been proposed for a networked or distributed environment. Early systems included ASAX (Mouinji, 1995), DIDS (Snapp, 1999) and NSTAT (Kemmerer, 1997). These systems require the audit data collected from different places to be sent to a central location for an analysis. NetSTAT (Vigna, 1999) is another example of such a system. In NetSTAT attack scenarios are modeled as hypergraphs and places are probed for network activities. Although NetSTAT also collects information in a distributed manner, it analyses them in a central place. The scalability of such systems is limited due to their centralized nature. To improve scalability later systems such as EMERALD (Porras, 1997), GriDS (Staniford, 1996) and AAFID (Spafford, 2000), deployed instruction detection systems at different locations and organized them into a hierarchy such that low-level IDSs send designated information to higher level IDSs. EMERALD uses both misuse detection and statistical anomaly detection techniques. This system employs a recursive framework, which allows generic components to be deployed in a distributed manner (Porras, 1997). To detect intruders, GriDS aggregates computer and network information into activity graphs which reveal the causal structure of network activity (Staniford, 1996). AAFID consists of agents, filters transceivers and monitors organized in a tree structure (Spafford, 2000). The hierarchical approaches employed by theses schemes scale better than the previous centralized approached. However, the main problem with such an approach is that if two or more IDSs that are far apart in the hierarchy detect a common intruder, the two detection cannot be correlated until the messages from the different IDSs reach a common high-level IDS. This will require the messages to traverse multiple IDSs resulting in communication overheads. The Common Intrusion Detection Framework (CIDF) (Staniford, 1998) goes one step further as it aims to enable different intrusion detection and response components to interoperate and share information and resources in a distributed environment. The intrusion detection inter-component adaptive negotiation protocol helps cooperating CIDF components to reach an agreement on each other's needs and capabilities (Feiertag, 2000). MADAM ID uses CIDF to automatically get audit data, build models, and distribute signatures for novel attacks so that the gap between the discovery and detection of new attacks can be reduced (Lee, 2000). The coordinated and response system (CARDS) (Ning, 2002) aims at detecting distributed attacks that cannot be detected using data collected from any single location. CARDS decomposes global representations of distributed attacks into smaller units that correspond to the distributed events

indicating the attacks. It then executes and coordinates the decomposed smaller units in the places where the corresponding units are observed. The message transmission between component IDSs is not determined by a centralized or hierarchical scheme, Instead, in CARDS, one component IDS sends a message to another only when the message is required by the later IDS to detect certain attacks. The communication cost is therefore reduced. Although JiNao (Jou, 2000) has been proposed as a distributed IDS for detecting intrusions network routing protocols, no specific mechanisms have been provided for doing so. JiNao focuses on the Open Shortest Path First (OSPF) protocol.

Software agents have been proposed as a technology for intrusion detection applications. Rationale for considering agents in an IDS ranges from increased adaptability for new threats to reduced communication costs. Since agents are independently executing entities, there is the potential that new detection capabilities can be added without completely halting, rebuilding, and restarting the IDS. Other potential advantages are described in (Jansen, 1999), (Kruegel, 2001). Kruegel and Toth (Kruegel, 2001) also identify downside tradeoffs including increased design and operational complexity. The Autonomous Agents for Intrusion Detection (AAFID) framework (Spafford, 2000) employs autonomous agents for data collection and analysis. AAFID utilizes agents hosted on network nodes, filters to extract pertinent data, transceivers to oversee agent operation, and monitors to receive reports from transceivers. These entities are organized into a hierarchical architecture with centralized control. Cooperating Security Managers (CSMs) (White, 1996) enable individual distributed intrusion detection packages to cooperate in performing network intrusion detection without relying on centralized control. Each individual CSM detects malicious activity on the local host. When suspicious activity is detected, each CSM will report any noteworthy activity to the CSM on the host from which the connection originated. The local CSM will not notify all networked systems, but rather only the system immediately before it in the connection chain. Other agent-based hierarchical architectures include the Intelligent Agents for Intrusion Detection project (Helmer, 1998) with a centralized data warehouse at the root, data cleaners at the leaves, and classifier agents in between. Bernardes and dos Santos Moreira (Bernardes, 2000) have proposed a hybrid framework with partially distributed decision making under the control of a centralized agent manager. Agents are deployed to observe behavior of the system and users. Agents communicate via messages to advise peers when an action is considered suspect. When an agent considers an activity to be suspect, an agent with a higher level of specialization for the suspected intrusion is activated. Agents then report their findings to a centralized manager. The main drawbacks with these systems, is that the use of one or more centralized repositories

leave at least some portion of the network exposed to malicious attacks including tampering and denial of service attacks. Even if an autonomous mobile decision-making agent was to detect a problem, interlocking mechanisms would be necessary to preclude any accidental or malicious removal, delay, or spoofing the agent. The Tethered Agent and Collective Hive (TACH) architecture includes a centralized Hive to keep track of agents and collected data and an Agent Registry (AR) to track fingerprints of agents (Lu, 2000). An Aglet-based framework for TACH incorporates mobile agents for virus detection and misuse detection (Kapoor, 2000). Limitations of TACH include the use of a centralized entity for agent control and a period communication protocol between agents with time-out detection used to detect status changes in the agents. If the centralized entity is disabled then the entire TACH system has been compromised. The distributed intrusion detection system (DIDS) (Mukherjee, 1994) used a combination of host and LAN monitors to observe system and network activity. A centralized director obtained information from the monitors to detect intrusions. The CIDF nomenclature mentioned above includes reconnaissance agents for data gathering, analysis agents, and decision-response agents (Staniford, 1998). The Computer Immunology Project at the University of New Mexico (Forrest, 1997) explored designs of IDSs based on ideas gleaned by examining animal immune systems. Small, individual agents would roam a distributed system, identify intrusions, and resolve the intrusions. One portion of the project developed a sense of self for security-related computer programs by observing the normal sets of system calls executed by the programs. This sense of self can be used to detect intrusions by discovering when a program executes an unusual set of system calls. The JAM Project at Columbia University (Stolfo, 1997) uses intelligent, distributed Java agents and data mining to learn models of fraud and intrusive behavior that can be shared between organizations. Helmar et al. propose lightweight agents for intrusion detection (Helmer, 2003). Their multi-agent system includes agents that travel between monitored systems in a network of distributed systems, obtain information from data cleaning agents, classify and correlate information, and report the information to a user interface and database via mediators. Agent systems with lightweight agent support allow runtime addition of new capabilities to agents. DeMara et. al. (DeMara, 2004) propose an IDS based on mobile agents for detecting malicious activity by people with legitimate access to resources and services. These include attacks such as spoofing, termination, sidetracking, alteration of internal data, and selective deception. Their system employs techniques such as encapsulation, redundancy, scrambling, and mandatory obsolescence.

DIDS are simply a superset of the conventional IDS implemented in a distributed environment. Due to the distributed nature the implementation poses several challenges. IDS could be embedded inside agents and placed in the network to be monitored. The individual IDS may be configured to detect a single attack, or they may detect several types of attacks. Each network component may host one or many IDS. Since there will be a large number of flag generators (detection of an attack, event etc.), these must be abstracted, analyzed, and condensed by a suitable architecture before arriving a final conclusion. Very often there would be a centralized analyzing and control facility. Some of the popular architectures are depicted in Figures 2-4. The Master-slave architecture is depicted in Figure 2. This architecture may be suitable for small networks. In a hierarchical architecture analysis and control are being done at different layers mainly because of the geographical distribution or due to the size of the network. Attacks/event detection information is passed to analyzer/controller nodes that aggregate information from multiple IDS agents. It is to be noted that the event information, which is detected by the IDS agents will follow a bottom up approach for analysis and the various command and control flow will follow a top-down approach. The physical location of IDS agents will be fixed since they monitor fixed network segments. In the case of hierarchical architecture, the analyzer/controller nodes may exist at many locations in the network since they receive their input and give their output via network connections. Depending on the network environment the communication between the different layers could be implemented as depicted in Figure 4.

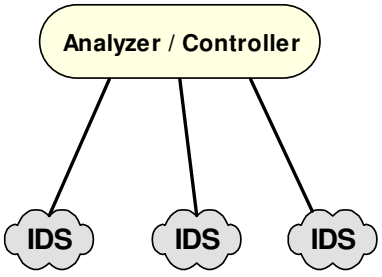

Figure 2. DIDS master-slave architecture

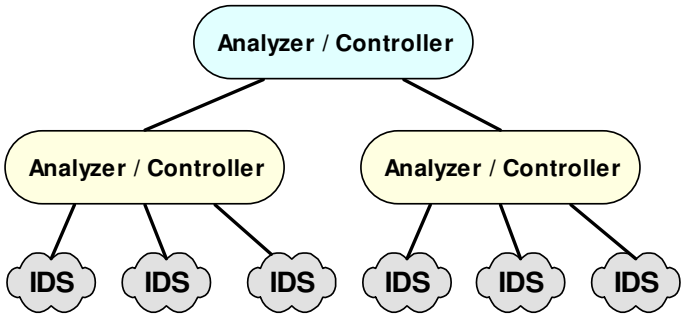

Figure 3. DIDS hierarchical architecture

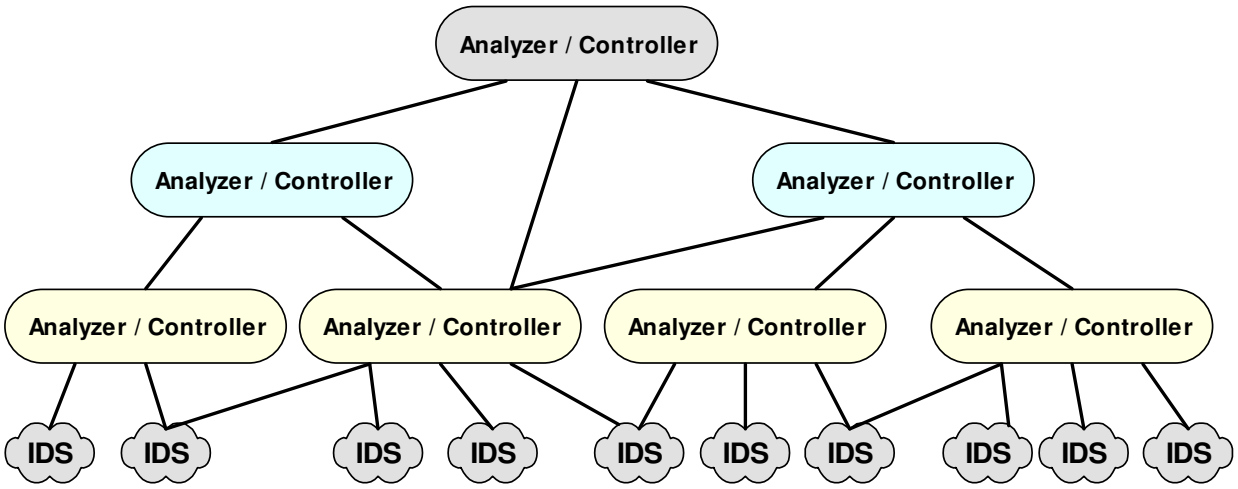

Figure 4. Hierarchical architecture with free communication between layers.

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

Agent modules running on the host machines are capable of data analysis and to formulate adequate response action and are very often implemented as read only and fragile. In the event of tampering or modification the agent reports to the server agent and automatically ends its life. Agents residing in the individual analyzer/controllers consist of modules responsible for agent regeneration, dispatch, updating and maintaining intrusion signatures and so on.

These agents control the individual IDS agents for monitoring the network, manage all the communication and life cycle of the IDS agents and also updates the IDS agents with detection algorithms, response and trace mechanisms.

4. Reducing the Data Features for Intrusion Detection Systems

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. Complex relationships exist between the features, which are difficult for humans to discover. IDS must therefore reduce the amount of data to be processed. This is very important if real-time detection is desired. Reduction can occur in one of several ways. Data that is not considered useful can be filtered, leaving only the potentially interesting data. Data can be grouped or clustered to reveal hidden patterns; by storing the characteristics of the clusters instead of the data, overhead can be reduced. Finally, some data sources can be eliminated using feature selection (Chebrolu, 2004).

4.1. 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, Θ) is a Directed Acyclic Graph (DAG) (N, A) where each node $n \in N$ represents a domain variable (e.g. a dataset attribute or variable), and each arc $a \in A$ between nodes represents a probabilistic dependency among the variables, quantified using a conditional probability distribution (CP table) $\theta_i \in \Theta$ for each node n_i. 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, lack of publicly available learning tools. In order to resolve these problems, two algorithms 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) 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. 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 dataset 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. Several other advantages of Bayesian networks are 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 framework. Also Bayes networks offer good generalization with limited training data, easy maintenance when adding new features or new training data.

Markov Blanket (MB) of the output variable *T,* is a novel idea for significant feature selection in large data sets (Tsamardinos, 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. This subset of nodes shields *n* from being affected by any node outside the blanket. When using a BN classifier on complete data, the Markov blanket of the class node forms feature selection and all features outside the Markov blanket are deleted from the BN.

4.2. Decision Tree Approach for Data Reduction

The Classification and Regression Trees (CART) methodology is technically called as binary recursive partitioning (Breiman, 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 tree is complete and assigning a class outcome to each terminal node. As an example, for the DARPA intrusion data set (KDD, 1999) with 5092 cases and 41 variables, CART considers up to 5,092 times 41 splits for a total of 208,772 possible splits. For splitting, 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. Instead of attempting to decide whether a given node is terminal or not, CART proceeds by growing trees until it is not possible to grow them any further. Once CART 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 made to the construction of the decision tree. Feature importance is determined by the role of each input variable either as a main splitter or as a surrogate. Surrogate splitters are defined as back-up rules that closely mimic the action of primary splitting rules. Suppose that, in a given model, the algorithm splits data according to variable '*protocol_type*' and if a value for '*protocol_type*' is not available, the algorithm might substitute *'service'* as a good surrogate. Variable importance, for a particular variable is the sum across all nodes in the tree of the improvement scores that the predictor has when it acts as a primary or surrogate (but not competitor) splitter. Example, for node *i*, if the predictor appears as the primary splitter then its contribution towards importance could be given as $i_{importance}$. But if the variable appears as the n^{th} surrogate instead of the primary variable, then the importance becomes $i_{importance} = (p^n) * i_{improvement}$ in which *p* is the 'surrogate improvement weight' which is a user controlled parameter set between (0-1).

4.3. Experiment Setup and Results

The data for our experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Labs (KDD, 1999). 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 contains 744 MB data with 4,940,000 records. The data set contains 24 attack types. All these attacks fall into four main categories (KDD, 1999).

1. **Denial of Service (DOS):** In this type of attacks 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 attacks an attacker who does not have an account on 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 vulnerability 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 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. 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 past two seconds 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 two seconds 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 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. 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, training phase and 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 constructs a model using the training data to give maximum generalization accuracy on the unseen data. The test data is then passed through the saved trained model to detect intrusions in the testing phase. The data set for our experiments contains randomly generated 11982 records having 41 features (KDD, 1999). The 41 features are labeled in order as *A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R,*

S, T, U, V, W, X, Y, Z, AA, AB, AC, AD, AF, AG, AH, AI, AJ, AK, AL, AM, AN, AO and the class label is named as *AP*. 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 are trained and tested with the same set of data. As the data set has five different classes we perform a 5-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. We selected the important features using the Markov blanket model and found out that 17 variables of the data set forms the Markov blanket of the class node as explained in Section 5.1. These 17 variables are *A, B, C, E, G, H, K, L, N, Q, V, W, X, Y, Z, AD* and *AF*.

4.3.1. Modeling IDS Using Bayesian Classifier

Furthermore, Bayesian network classifier is constructed using the training data and then the classifier is used on the test data set to classify the data as an attack or normal. Table 1 depicts the performance of Bayesian belief network by using the original 41 variable data set and the 17 variables reduced data set. The training and testing times for each classifier are decreased when 17 variable data set is used. Using the 17 variable data set there is a slight increase in the performance accuracy for *Normal* class compared to the 41 variable data set.

Table 1. Performance of Bayesian Belief Network

Attack Class	41 variables			17 variables		
	Train (sec)	Test (sec)	Accuracy (%)	Train (sec)	Test (sec)	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

4.3.2. Modeling IDS Using Decision Trees

We decided the important variables depending on the contribution of the variables for the construction of the decision tree. Variable rankings were generated in terms of percentages. We eliminated the variables that have 0.00% rankings and considered only the primary splitters or surrogates as explained in Section 4.1. 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.

Table 2. Performance of classification and regression trees

Attack Class	41 variable data set			12 variable data set		
	Train (sec)	Test (sec)	Accuracy (%)	Train (sec)	Test (sec)	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**

Further the classifier is constructed using the training data and then the test data is passed through the saved trained model. Table 2 compares the performance of CART using the 41 variable original data set and the 12 variable reduced data set. Normal class is classified 100 percent correctly. Furthermore, the accuracies of classes U2R and R2L have increased by using the 12 variable reduced data set. It is also found that CART could classify accurately on smaller data sets.

4.3.3. Modeling IDS Using Bayesian Networks and Decision Trees with Reduced Dataset

Further, we used the Bayesian reduced 17 variable data set to train CART and the CART reduced 12 variable dataset to train Bayesian network. As illustrated in Table 3 except R2L all other classes were classified well by the CART algorithm. Moreover, training and testing time for each class are greater for Bayesian network classifier compared to CART algorithm.

Table 3. Performance of Bayesian and CART using reduced datasets

Attack Class	Bayesian with 12 variables			CART with 17 variables		
	Train (sec)	Test (sec)	Accuracy (%)	Train (sec)	Test (sec)	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

4.3.4. Ensemble Approach Using Reduced Data Sets

Empirical observations show that different classifiers provide complementary information about the patterns to be classified. Although for a particular problem 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 single classifier for decision, instead different classifiers individual information 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. Various techniques are developed for the ensemble approach. One technique is to use different training models for different base classifiers and then combine their outputs, another one uses different subset of features for different base classifiers and combines their outputs. In this approach we use the same data set as well as feature set for all the base classifiers and combine them to give the final output of the ensemble approach. We first construct the Bayesian network classifier and CART models individually to obtain a very good generalization performance. 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 obtained for the individual classifiers. 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. When the highest scored class fails, then the next highest scored class is given preference. Table 5 illustrates the ensemble results using the different data sets. From the results, we can conclude that ensemble approach gives better performance than the two individual separately used models. The ensemble approach basically exploits the differences in misclassification (by individual models) and improves the overall performance. Figure 5 illustrates the developed hybrid IDS model to detect the different attacks 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 80% and 99.47% accuracies respectively.

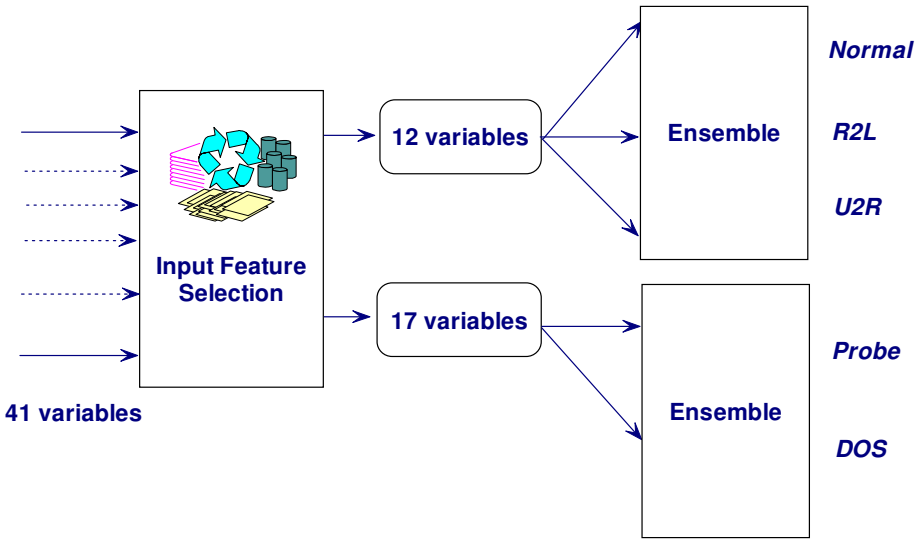

Figure 5. Developed ensemble IDS model for different attack classes

Table 5. 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

5. Improving the Performance (Accuracy) of Intrusion Detection Systems

5.1. Hybrid Support Vector Machines (SVM) – Decision Trees (DT) Approach

A 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 a SVM leads to 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. Kernel transforms linear algorithms into nonlinear ones via a map into feature spaces. 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.

Decision Trees (DT) induction is one of the classification algorithms in the data mining. 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 $X \in Z$. 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. Decision tree classifies the given data item using the values of its attributes. The decision tree is initially constructed from a set of pre-classified data. Each data item is defined by values of the attributes. The main approach 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 current subset belongs to the same class. A decision tree consists of nodes, leaves and edges. A node of a decision tree 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, which are labeled according to 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 decision tree uses the training data, which is described in terms of the attributes.

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, 2003). The overall functioning of the system depends on the correct functionality of all the layers. Figure 5 shows the architecture of the hybrid intelligent system with decision tree and SVM.

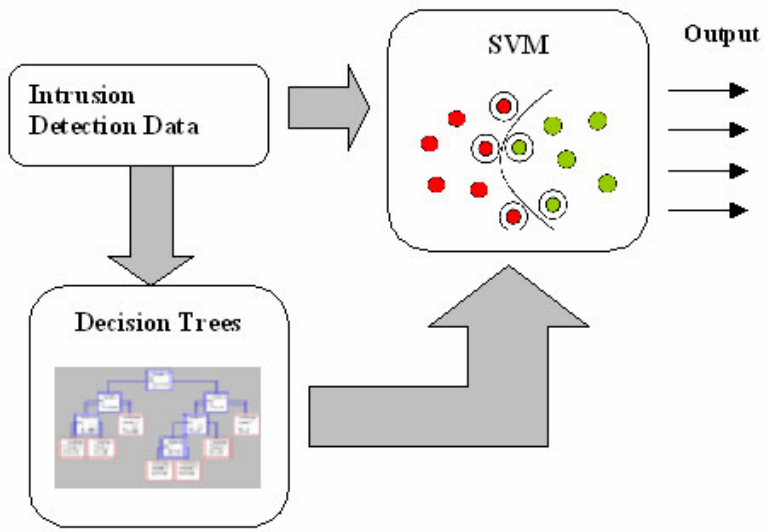

Figure 6. Hybrid decision tree-SVM model

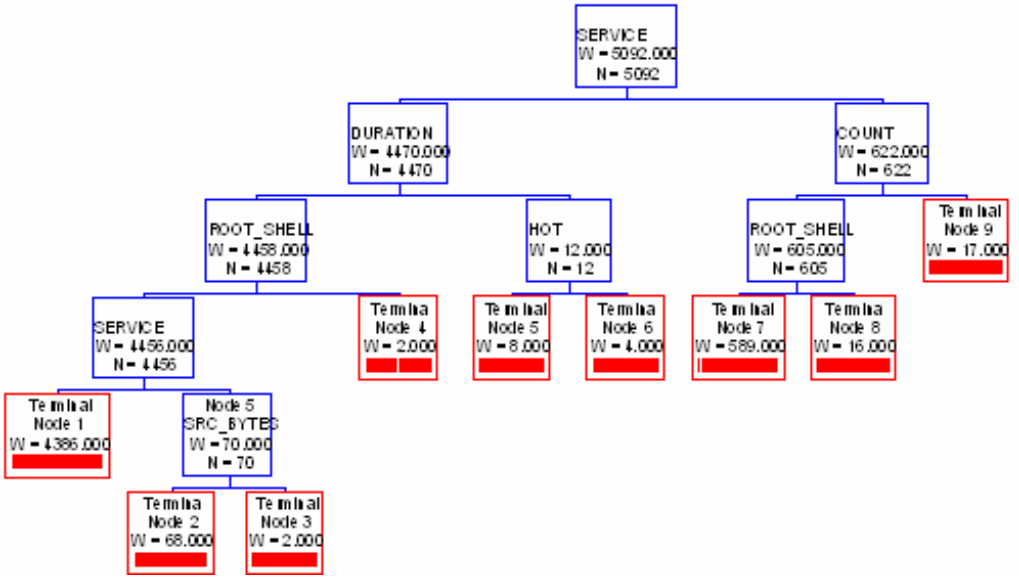

Figure 7. Decision tree output structure showing the terminal nodes

The data set is first passed through decision tree and node information is generated. Node information is determined according to the rules generated by the decision tree. Terminal nodes are numbered left to right starting with 1 as shown in the Figure 7. All the data set records are assigned to one of the terminal nodes, which represent the particular class or subset. 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 DT could improve the performance of SVM (Peddabachigari, 2004).

5.2. Ensemble Approach

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 output of the classifier scores were calculated. Example, for class 1 if the DT works best, followed by the DT-SVM and SVM model, the decision tree is assigned highest weight, followed by the hybrid decision tree-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 one is declared the winner and is the actual output of the ensemble approach. The architecture of the ensemble approach is depicted in Figure 8.

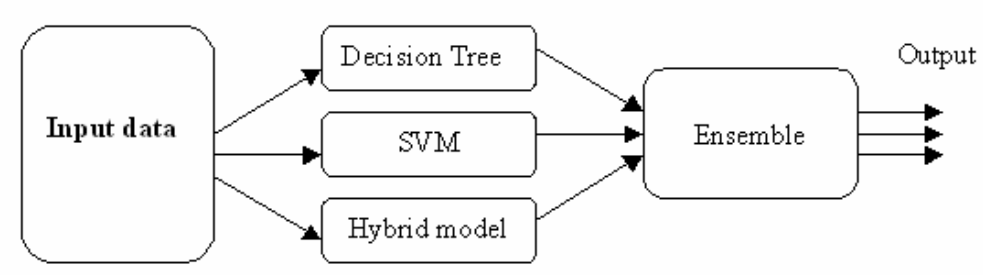

Figure 8. Architecture of ensemble approach

5.3. Experiment Setup and Performance Evaluation

The same data set used for the data reduction phase is used in our experiments but all the 41 features were used for the experiments. Experiments have two phases namely 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. Experiments were performed using an AMD Athlon, 1.67 GHz processor with 992 MB of RAM.

5.3.1. Support Vector Machines

Kernel option defines the feature space in which the training set examples will be classified. The polynomial kernel was used for our experiments. We observed that for different class of data, different polynomial degrees gives different performance and the results are presented in the Table 6.

Table 6. Classification accuracy of 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

5.3.2. Hybrid Decision Tree-SVM

A Hybrid decision tree-SVM model has two steps for constructing the classifier. The data sets were first passed through the decision tree 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 decision tree-SVM. Performance of hybrid decision tree-SVM is illustrated in Table 7. Hybrid decision tree-SVM works better than both decision tree and SVM for normal class. For U2R and R2L classes it performed better than a direct SVM approach. From the above results we can conclude that although the node information generated by the decision tree enhances the performance of SVM but on the whole the hybrid decision tree-SVM model did not give the expected performance.

5.3.3. Ensemble Approach

In this approach first we construct decision tree, SVM and hybrid decision tree-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 7. Empirical results depict that the proposed ensemble approach gives better performance for detecting probes and U2R attacks than all the three individual models.

Table 7. 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	97.85	98.57	98.57	**100.00**
DOS	99.47	**99.92**	99.92	99.92
U2R	48.00	40.00	48.00	**68.00**
R2L	90.58	33.92	37.80	**97.16**

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 basically exploits the differences in misclassification and improves the overall performance. To minimize the computational load we further propose an IDS model as depicted in Figure 9. The hybrid IDS model makes uses of individual, hybrid and ensemble approaches to maximize the computational efficiency and detection accuracy.

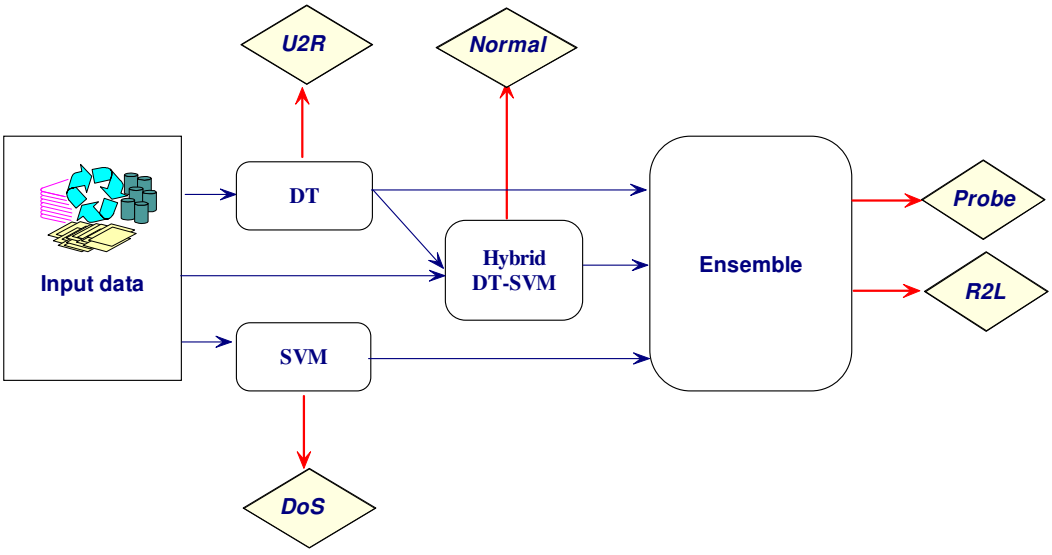

Figure 9. IDS based on a hierarchical intelligent system

6. Conclusions

Effective intrusion detection and management systems are critical components of homeland security as they are in the forefront of the battle again cyber-terrorism. In this chapter we presented a framework for Distributed Intrusion Detection Systems (DIDS) with a focus on improving the intrusion detection performance by reducing the input features and hybrid approaches for combining base classifiers. We used the feature selection method using Markov blanket model and decision tree analysis. Following this, we explored 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.

In the next phase we illustrated the importance of hybrid approaches by designing a hybrid DT-SVM model and an ensemble approach with decision tree, SVM and DT-SVM models as base classifiers. Empirical results reveal that the hybrid decision tree-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 U2R 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 the hybrid IDS model to make use of the best performances delivered by the individual base classifiers and the ensemble approach.

With the increasing incidents of cyber attacks, building an effective intrusion detection models with good accuracy and real-time performance are essential. This field is developing continuously. More data mining techniques should be investigated and their efficiency should be evaluated as intrusion detection models.

Acknowledgements

The authors wish to acknowledge the contributions (Peddabachigari, 2004; Chebrolu, 2004) made by Sandhya Peddabachigari and Srilatha Chebrolu who conducted the experiments reported in this paper during their graduate studies at Oklahoma State University, USA. Authors are grateful to the anonymous reviewers for the technical comments which helped to improve the clarity of the paper.

References

(Abraham, 2003) A Abraham, Intelligent Systems: Architectures and Perspectives, Recent Advances in Intelligent Paradigms and Applications, Abraham A., Jain L. and Kacprzyk J. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter 1, pp. 1-35, 2003.

(Bernardes, 2000) M C Bernardes and E dos Santos Moreira, "Implementation of an intrusion detection system based on mobile agents", Proceedings International Symposium on Software Engineering for Parallel and Distributed Systems, pp. 158-164, 2000

(Breiman, 1984) L Breiman, J Friedman, R Olshen, C Stone, Classification and Regression Trees, Chapman and Hall, New York 1984

(Chebrolu, 2004) S Chebrolu, A Abraham and J Thomas, Feature Deduction and Ensemble Design of Intrusion Detection Systems, Computers and Security, Elsevier Science, 2004 (in press).

(Denning, 1997) E. Denning. "An Intrusion Detection Model". In IEEE Transactions on Software Engineering, pp. 222-228, February 1997.

(Debar, 1992) M Debar, DBecke, and A Siboni. "A Neural Network Component for an Intrusion Detection System". Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, 1992.

(DeMara, 2004) Ronald F. DeMara, and Adam J. Rocke, "Mitigation of network tampering using dynamic dispatch of mobile agents", Computers & Security, Vol. 23, pp. 31 – 42, 2004

(Dunigan, 1999) T Dunigan and G Hinkel, "Intrusion detection and intrusion prevention on a large network: A case study", Proc. of workshop on intrusion detection and network monitoring, 1999

(Feiertag, 2000) R Feiertag, S Rho, L Benzingher, S Wu, T Redmond, C Zhang, K Levitt, D Peticolas, M Heckman, S Staniford, J McAlerney, "Intrusion Detection Inter-Component Adaptive Negotiation". Computer Networks, Vol. 34, pp. 605-621, 2000

(Forrest, 1997) S Forrest, S A Hofmeyr, A Somayaji, "Computer immunology", CACM 40 (10), 88–96.1997

(Friedman, 1991) J H Friedman, Multivariate Adaptative Regression Splines. Annals of Statistics Vol. 19 1991

(Garvey, 1991) T. D. Garvey and T. F. Lunt, "Model based intrusion detection". In Proceedings of the 14th National Computer Security Conference, pages 372-385, October 1991.

(Helmer, 1998) G Helmer, J Wong, V Honavar and L Miller, "Intelligent agents for intrusion detection", 1998,Available from http://citeseer.nj.nec.com/helmer98intelligent.html.

(Helmer, 2003) Guy Helmer, Johnny S.K. Wong, Vasant Honavar, Les Miller, Yanxin Wang, "Lightweight agents for intrusion detection", The Journal of Systems and Software Vol. 67, pp. 109–122, 2003

(Ilgun, 1993) K Ilgun, "USTAT: A Real-time Intrusion Detection System for UNIX," 16-28. Proceedings of the 1993 Computer Society Symposium on Research in Security and Privacy. Oakland, California, May 24-26, 1993. Los Alamitos, CA: IEEE Computer Society Press, 1993.

(Kapoor, 2000) B Kapoor, "Remote misuse detection system using mobile agents and relational database query techniques", Master's thesis, University of Central Florida, 2000.

(KDD, 1999) KDD cup 99 Intrusion detection data set, Retrieved October 25, 2004, from <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz>

(Kemmerer, 1997) R A Kemmerer, “NSTAT: a Model-based Real-time Network Intrusion Detection System”, Technical Report TRCS97-18, Reliable Software Group, Department of Computer Science, University of California at Santa Barbara, 1997

(Kumar, 1994) S. Kumar and E. H. Spafford. ”An Application of Pattern Matching in Intrusion Detection”. Technical Report CSD-TR-94-013, Purdue University, 1994.

(Kumar, 1995) S. Kumar. ”Classification and Detection of Computer Intrusions”. PhD Thesis, Department of Computer Science, Purdue University, August 1995.

(Jansen, 1999) W Jansen, P Mell, T Karygiannis and D Marks, “Applying mobile agents to intrusion detection and response”, National Institute of Standards and Technology, Computer Security Division; 1999. Available from:http://csrc.nist.gov/publications/ nistir/ir6416.pdf.

(Jou, 2000) Y F Jou, F Gong, C Sargor, X Wu, S F Wu, H C Chang, F Wang, “Design and implementation of a scalable intrusion detection system for the protection of network infrastructure”, DARPA Information Survivability Conference and Exposition, 2000

(Kruegel, 2001) C Kruegel and T Toth, “Applying mobile agent technology to intrusion detection”, Proc. ICSE Workshop on Software Engineering and Mobility; 2001. Available from: http://citeseer. nj.nec.com/kr01applying.html.

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

(Lee, 2000) W Lee, R A Nimbalker, K K Yee, S B Patil, P H Desai, P P Tran, S J Stolfo, “A data mining and CIDF based approach for detecting novel and distributed intrusions”, Proc. 3rd International Workshop on Recent Advances in Intrusion Detection, 2000

(Lu, 2000) J Lu, “Mobile agent protocols for distributed detection of network intrusions”, Master’s thesis, University of Central Florida, 2000.

(Mouinji, 1995) A Mouinji, B L Charlier, D Zampunieris, N Habra, “Distributed Audit Trail Analysis”, Proceedings of the ISOC 95 Symposium on Network and Distributed System Security”, pp. 102-112, 1995

(Mukherjee, 1994) B Mukherjee, L Todd Heberlein, K N Levitt, 1994. “Network intrusion detection. IEEE Network, Vol. 8, No. 3, pp.26–41, 1994

(Mukkamala, 2003) S. Mukkamala, A.H. Sung and A. Abraham, ”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, Springer Verlag, Germany, pp. 239-248, 2003.

(Mukkamala, 2004a) S. Mukkamala, A.H. Sung and A. Abraham, ”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, Robert Orchard, Chunsheng Yang, Moonis Ali (Eds.), Lecture Notes in Computer Science 3029, Springer Verlag, Germany, pp. 633-642, 2004.

(Mukkamala, 2004b) S. Mukkamala, A.H. Sung, A. Abraham and V. Ramos, ”Intrusion Detection Systems Using Adaptive Regression Splines”, 6th International Conference on Enterprise Information Systems, ICEIS'04, Portugal, I. Seruca, J. Filipe, S. Hammoudi and J. Cordeiro (Eds.), ISBN 972-8865-00-7, Vol. 3, pp. 26-33, 2004.

(Ning, 2002) P Ning, S Jajodia, X S Wang, “Design and implementation of a decentralized prototype system for detecting distributed attacks”, Computer Communications, Vol. 25, pp. 1374-1391, 2002

(Peddabachigari, 2004) S Peddabachigari, A Abraham and J Thomas, Intrusion Detection Systems Using Decision Trees and Support Vector Machines, International Journal of Applied Science and Computations, USA, 2004 (in press).

(Porras, 1992) P A Porras. "STAT: A State Transition Analysis Tool for Intrusion Detection". Master's Thesis, Computer Science Dept., University of California, Santa Barbara, July 1992.

(Porras, 1997) P A Porras, P G Neumann, "EMERALD: event monitoring enabling response to anomalous live disturbances", Proceedings 20th National Information Security Conference, NIST 1997

(Shah, 2004) K. Shah, N. Dave, S. Chavan, S. Mukherjee, A. Abraham and S. Sanyal, "Adaptive Neuro-Fuzzy Intrusion Detection System", IEEE International Conference on Information Technology: Coding and Computing (ITCC'04), USA, IEEE Computer Society, Volume 1, pp. 70-74, 2004.

(Snapp, 1999) S R Snapp, J Bretano, G V Diaz, T L Goan, L T Heberlain, C Ho , K N Levitt, B Mukherjee, S E Smaha, T Grance, D M Teal, D Mansur, "DIDS (Distributed Intrusion Detection System) – motivation architecture and an early prototype", Proceedings 14th National Computer Security Conference, Washington DC, October, pp. 167-176, 1999

(Spafford, 2000) E H Spafford, D Zamboni, "Intrusion detection using autonomous agents", Computer Networks, 34, pp. 547-570, 2000

(Staniford, 1996) S Staniford-Chen, S Cheung, R Crawford, M Dilger, J Frank, J Hoagland, K Levitt, C Wee, R Yipi, D Z Erkle, "GriDS – a large scale intrusion detection system for large networks", Proceedings 19th National Information Security Conference, Vol. 1, pp. 361-370, 1996

(Staniford, 1998) S Staniford-Chen, S, B Tung, and D Schnackenberg, "The Common Intrusion Detection Framework (CIDF)". Proc. Information Survivability Workshop, Orlando FL, October 1998.

(Stolfo, 1997) S J Stolfo, A L Prodromidis, S Tselepis, W Lee, D Fan, P K Chan, "JAM: Java agents for meta-learning over distributed databases. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, USA, pp. 74–81.1997

(Teng, 1990) H. S. Teng, K. Chen and S. C. Lu. "Security Audit Trail Analysis Using Inductively Generated Predictive Rules". In Proceedings of the 11th National Conference on Artificial Intelligence Applications, pages 24-29, IEEE, IEEE Service Center, Piscataway, NJ, March 1990.

(Tsamardinos, 2003) Tsamardinos I., Aliferis C.F.and Statnikov A., "Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations", 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, ACM Press, pages 673-678, 2003.

(Vapnik, 1995) V.N. Vapnik, The Nature of Statistical Learning Theory. Springer, 1995.

(Vigna, 1999) G Vigna, R A Kemmerer, "NetSTAT: A network-based intrusion detection system", Journal Computer Security, Vol. 7, No, 1, pp. 37-71, 1999

(Wiser, 2002), L Wiser, Intrusion Detection and Homeland Security, Ask the Expert, Retrieved October 25, 2004, from http://www2.cio.com/ask/expert/2002/questions/question1522.html?CATEGORY=15&NAME=Internet

(White, 1996) G B White, E A Fisch and U Wpooch, "Cooperating security managers: a peer-based intrusion detection system", IEEE Networks, 1996

Author Biographies

Ajith Abraham is currently a Distinguished Visiting Professor at Chung-Ang University, Korea. He received PhD degree from 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. 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.

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).