Contents

Chapter 1
Introduction

In this thesis, we have explained two of the latest hybrid intrusion detection techniques which address three significant issues which severely restrict the utility of anomaly and hybrid intrusion detection systems in present networks and applications. The three issues are: limited attack detection coverage, large number of false alarms and inefficiency in operation.

Today's intrusion detection systems have a limited attack detection coverage due to which it may suffer from some new attack. The number of false alarms may put the network administrator to completely ignore them completely. As the most of existing intrusion detection systems such as Snort and others are developed using knowledge engineering approaches where domain experts can build focused and optimized pattern matching models. Though such systems result in very few false alarms, they are specific in attack detection and often tend to be incomplete. As a result their effectiveness is limited. Further, due to their manual development process, signature based systems are expensive and slow to build. We, thus, address these shortcomings and develop better anomaly and hybrid intrusion detection systems which are accurate in attack detection, efficient in operation and have wide attack detection coverage.

1.1 Motivation and Problem Description

Intrusion detection as defined by the Sysadmin, Audit, Networking, and Security (SANS) institute is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource [2]. Due to the rapid growth of internet and usage, the number of attacks has increased. Attackers exploit the vulnerabilities in the system to attack the system. So, sophisticated tool are required to protect these applications and data. The major objective of the intrusion detection systems is to provide an extra layer of security to these application and data.

Increasing dependence of businesses on the services over the Internet, though, has led to their rapid growth; it has also made the networks and applications a prime target of attacks.

Configuration errors and vulnerabilities in software are exploited by the attackers who launch powerful attacks such as the Denial of Service (DoS) and Information attacks. Rapid increase in the number of vulnerabilities has resulted in an exponential rise in the number of attacks. According to the Computer Emergency Response Team (CERT), the number of vulnerabilities in software has been increasing and many of them exist in highly deployed software [3], [4].

Considering that it is near to impossible to build 'perfect' software, it becomes critical to build effective intrusion detection systems which can detect attacks reliably. The prospect of obtaining valuable information, as a result of a successful attack, subside the threat of legal convictions. The inability to prevent attacks furthers the need for intrusion detection. The problem becomes more profound since authorized users can misuse their privileges and attackers can masquerade as authentic users by exploiting vulnerable applications. Given the diverse type of attacks (Denial of Service, Probing, Remote to Local, User to Root and others), it is a challenge for any intrusion detection system to detect a wide variety of attacks with very few false alarms in real-time environment. Ideally, the system must detect all intrusion with no false alarms.

1.1.1 Research Objectives

In this thesis:

1. Our aim is to analyze the hybrid intrusion detection techniques which have broad attack detection coverage and which are not specific in detecting only the previously known attacks.
2. Our aim is to analyze the techniques which reduce the number of false alarms improving their attack detection accuracy.
3. Our aim is to analyze the techniques which work efficiently on high speed networks.

1.2 Contributions to Thesis

Attacker use the vulnerabilities present in the various network applications to launch new and unseen attacks. They normally use a sequence of events and may hide these events in normal events. It affects the classification and produces a large number of false alarms. It has been seen that anomaly based systems use the attack patterns and take features individually. So we are focusing on hybrid intrusion detection systems that use both normal and attack data to train the IDS.

We have implemented and compared two of the latest techniques Layered approach to intrusion detection using conditional random fields and Fuzzy cluttering with artificial neural networks for intrusion detection.

1.3 Thesis Organization

This thesis is organized as follows:

Chapter 2 describes the basics of intrusion detection and classification of the intrusion detection systems. It provides the related literature review.

Chapter 3 provides layered approach overview and compares it to the centralized systems.

Chapter 4 provides the details of the layered approach to intrusion detection using conditional random fields.

Chapter 5 provides another approach Fuzzy cluttering with artificial neural networks for intrusion detection.

Chapter 6 provides the implementation details with conclusion and future scope.

1.4 Terminology

Attack (or Intrusion): An attack can be defined as a violation of the security policy.

Authentication: Authentication refers to validating the identity of a user or application accessing a system.

Authorization: Authorization refers to a process to infer whether an authenticated user or an application has the privilege to access a particular resource.

Availability: Availability refers to the property of a system which must ensure that any resource (data or service) must be available to an authenticated and authorized entity upon request.

Conditional Random Fields: Conditional random fields are undirected and discriminative graphical models which directly model the conditional distribution of the labels, y, given the observation sequence x.

Confidentiality: Confidentiality refers to protecting the disclosure of data to unauthenticated and/or unauthorized entity.

Data Mining: Data mining is the process of finding patterns, automatically, generally from a very large data set.

Integrity: Integrity refers to ensuring that the data is free from modification by unauthenticated and/or unauthorized entity.

Intrusion Detection: Given, an attack is a violation of a security policy; intrusion detection is defined as the process of identifying such violations.

Intrusion Detection System: An intrusion detection system is a system which includes processes to detect, identify and respond to attacks. The processes include audit data collection, data pre-processing, data analysis and decision making followed by recovery and response mechanisms.

Machine Learning: Machine learning refers to the ability of a machine (usually a computational system) to learn from the data in order to build a system which can then be used for understanding underlying process that generated the data, classification of new instances, clustering and other artificial intelligence tasks.

Non-Repudiation: Non-Repudiation refers to the property which ensures accountability by preventing an entity from denying when it accessed a particular resource.

Pattern Matching: Pattern matching refers to the capability of a system to identify distinctive features which constitute a well known pattern.

Security Policy: A security policy is a policy document which defines the rules applicable to access any resource (data or service) in a network.

Chapter 2 Background

Intrusion detection as defined by the SysAdmin, Audit, Networking, and Security (SANS) Institute is the art of detecting inappropriate, inaccurate, or anomalous activity [2]. The cost involved in protecting valuable resources is often negligible as compared to the actual cost of a successful intrusion, which strengthens the need to develop more powerful intrusion detection systems. Thus, there is a need to safeguard the networks from known vulnerabilities and at the same time take steps to detect new and unseen, but possible, system abuses by developing more reliable and efficient intrusion detection systems. Any intrusion detection system has some inherent requirements. Its prime purpose is to detect as many attacks as possible with minimum number of false alarms, i.e., the system must be accurate in detecting attacks. However, an accurate system that cannot handle large amount of network traffic and is slow in decision making will not fulfill the purpose of an intrusion detection system. We desire a system that detects most of the attacks, gives very few false alarms, copes with large amount of data, and is fast enough to make real-time decisions. Intrusion detection started in 1980's and since then a number of approaches have been introduced to build intrusion detection systems. Various techniques such as association rules, clustering, naive Bayes classifier, support vector machines, genetic algorithms, artificial neural networks, and others have been applied to detect intrusions. However, intrusion detection is still at its infancy and naive attackers can launch powerful attacks which can bring down an entire network [5]. To identify the shortcoming of different approaches for intrusion detection, we explore the related research in intrusion detection. We describe the problem of intrusion detection in detail and analyze various well known methods for intrusion detection with respect to two critical requirements viz. accuracy of attack detection and efficiency of system operation. We observe that present methods for intrusion detection suffer from a number of drawbacks which significantly affect their attack detection capability. Then we explained two of the latest approaches layered approach using Conditional Random Field [67] and FCANN [68] for building intrusion detection systems which can operate efficiently and which can detect a wide variety of attacks with relatively higher accuracy, both at the network and at the application level. In the subsequent chapters, we will explain both the techniques in depth and compare the results.

2.1 Introduction

At present, networks use the concept of resource sharing as it is a necessity for collaboration, and provides an easy means of communication and economic growth. The systems are getting bigger with more and more add on features making them complex resulting in vulnerabilities in software. Ease of access of resources in addition to vulnerabilities and poor management of resources can be exploited to launch attacks [82]. As a result, the number of attacks has increased significantly [3]. Additionally, the attacks have become more complex and difficult to detect using traditional intrusion detection approaches, demanding more effective solutions [5]. More stringent monitoring has further increased the resources required by the intrusion detection systems. However, addition of more resources may not always provide a desired level of security.

The notion of intrusion started in 1980's with a paper from Anderson [6], which described that audit trails contain valuable information and could be utilized for the purpose of misuse detection by identifying anomalous user behaviour. The lead was then taken by Denning at the SRI International and the first model of intrusion detection, 'Intrusion Detection Expert System' (IDES) [7], was born in 1984. This further led to the concept of distributed intrusion detection system which augmented the existing solution by tracking client machines as well as the servers. The last system to be released under the same generation, called 'Stalker', was released in 1989 which was again a host based, pattern matching system [8]. Until then, the majority of the systems were host based and analyzed the individual host level audit records. Todd Heberlein, in 1990 introduced the concept of network intrusion detection and came up with the system called the 'Network Security Monitor' (NSM), [83]. These developments gradually paved way for the intrusion detection systems to enter into the commercial market with products such as 'Net Ranger', 'Real Secure' and 'Snort' acquiring big market shares [8].

Present intrusion detection systems are very often based on analyzing individual audit patterns by extracting signatures or are based on analyzing summary statistic collected at the network or at the application level , [24]. Such systems are unable to detect attacks reliably because they neglect the sequence structure in the audit patterns and consider every pattern to be independent. In most situations such independence assumptions do not hold which severely affect the attack detection capability of an intrusion detection system.

Another approach for intrusion detection is based on analyzing sequence structure in the audit patterns. Methods based on analyzing sequence of system calls issued by privileged processes are well known [11], [12]. However, to reduce system complexity, the system considers only one feature which is the sequence of system calls. Other features, such as the arguments of the system calls, are ignored. In cases, when multiple features are considered, the features are assumed independent and separate models are built using individual features. Results from all the models are then combined using a voting mechanism. This again may not detect attacks reliably. To improve attack detection, all of the features must be considered collectively and not independently [13], [14]. Assuming events to be independent makes the model simple and improves speed of operation; but at the cost of reduced attack detection and increased number of false alarms. Frequent false alarms, in turn, make the system administrators to ignore the alarms altogether.

Present networks and applications are, thus, far away from a state where they can be considered secure. Hence, in this chapter we explore the problem of intrusion detection to identify the root causes of the inability of the present intrusion detection systems to detect attacks reliably. We then motivate the use of conditional random fields [49] for building effective network and application intrusion detection systems.

2.2 Intrusion Detection and Intrusion Detection System

The intrusion detection systems are a critical component in the network security arsenal. Security is often implemented as a multi layer infrastructure and different approaches for providing security can be categorized into the following six areas [15]:

1. Attack Deterrence – Attack deterrence refers to persuading an attacker not to launch an attack by increasing the perceived risk of negative consequences for the attacker. Having a strong legal

system may be helpful in attack deterrence. However, it requires strong evidence against the attacker in case an attack was launched. Research in this area focuses on methods such as those discussed in [40] which can effectively trace the true source of attack as very often the attacks are launched with spoofed source IP address. (Spoofing refers to sending IP packets with modified source IP address so that the true sender of the packet cannot be traced.)

2. Attack Prevention – Attack prevention aims to prevent an attack by blocking it before an attack can reach the target. However, it is very difficult to prevent all attacks. This is because, to prevent an attack, the system requires complete knowledge of all possible attacks as well as the complete knowledge of all the allowed normal activities which is not always available. An example of attack prevention system is a firewall [16].

3. Attack Deflection – Attack deflection refers to tricking an attacker by making the attacker believe that the attack was successful though, in reality, the attacker was trapped by the system and deliberately made to reveal the attack. Research in this area focuses on attack deflection systems such as the honey pots [53].

4. Attack Avoidance – Attack avoidance aims to make the resource unusable by an attacker even though the attacker is able to illegitimately access that resource. An example of security mechanism for attack avoidance is the use of cryptography [18]. Encrypting data renders the data useless to the attacker, thus, avoiding possible threat.

5. Attack Detection – Attack detection refers to detecting an attack while the attack is still in progress or to detect an attack which has already occurred in the past. Detecting an attack is significant for two reasons; first the system must recover from the damage caused by the attack and second, it allows the system to take measures to prevent similar attacks in future. Research in this area focuses on building intrusion detection systems.

6. Attack Reaction and Recovery – Once an attack is detected, the system must react to an attack and perform the recovery mechanisms as defined in the security policy. Tools available to perform attack detection followed by reaction and recovery are known as the intrusion detection

systems. However, the difference between intrusion prevention and intrusion detection is slowly diminishing as the present intrusion detection systems increasingly focus on real-time attack detection and blocking an attack before it reaches the target. Such systems are better known as the Intrusion Prevention Systems.

2.2.1 Principles and Assumptions in Intrusion Detection

Denning [7] defines the principle for characterizing a system under attack. The principle states that for a system which is not under attack, the following three conditions hold true:

1. Actions of users conform to statistically predictable patterns.
2. Actions of users do not include sequences which violate the security policy.
3. Actions of every process correspond to a set of specifications which describe what the process is allowed to do.

Systems under attack do not meet at least one of the three conditions. Further, intrusion detection is based upon some assumptions which are true regardless of the approach adopted by the intrusion detection system. These assumptions are:

1. There exists a security policy which defines the normal and (or) the abnormal usage of every resource.

2. The patterns generated during the abnormal system usage are different from the patterns generated during the normal usage of the system; i.e., the abnormal and normal usage of a system results in different system behaviour. This difference in behaviour can be used to detect intrusions.

As we shall discuss later, different methods can be used to detect intrusions which make a number of assumptions that are specific only to the particular method. Hence, in addition to the definition of the security policy and the access patterns which are used in the learning phase of the detector, the attack detection capability of an intrusion detection system also depends upon the assumptions made by individual methods for intrusion detection [19].

2.2.2 Components of Intrusion Detection Systems

An intrusion detection system typically consists of three sub systems or components:

1. Data Preprocessor – Data preprocessor is responsible for collecting and providing the audit data (in a specified form) that will be used by the next component (analyzer) to make a decision. Data preprocessor is, thus, concerned with collecting the data from the desired source and converting it into a format that is comprehensible by the analyzer.

Data used for detecting intrusions range from user access patterns (for example, the sequence of commands issued at the terminal and the resources requested) to network packet level features (such as the source and destination IP addresses, type of packets and rate of occurrence of packets) to application and system level behavior (such as the sequence of system calls generated by a process.) We refer to this data as the audit patterns.

2. Analyzer (Intrusion Detector) – The analyzer or the intrusion detector is the core component which analyzes the audit patterns to detect attacks. This is a critical component and one of the most researched. Various pattern matching, machine learning, data mining and statistical techniques can be used as intrusion detectors. The capability of the analyzer to detect an attack often determines the strength of the overall system.

3. Response Engine – The response engine controls the reaction mechanism and determines how to respond when the analyzer detects an attack. The system may decide either to raise an alert without taking any action against the source or may decide to block the source for a predefined period of time. Such an action depends upon the predefined security policy of the network.

In [20], the authors define the Common Intrusion Detection Framework (CIDF) which recognizes a common architecture for intrusion detection systems. The CIDF defines four components that are common to any intrusion detection system. The four components are; Event generators (Eboxes), event Analyzers (A-boxes), event Databases (D-boxes) and the Response units (R-boxes). The additional component, called the D-boxes, is optional and can be used for later analysis.

2.2.3 Challenges and Requirements for Intrusion Detection Systems

The purpose of an intrusion detection system is to detect attacks. However, it is equally important to detect attacks at an early stage in order to minimize their impact. The major challenges and requirements for building intrusion detection systems are:

1. The system must be able to detect attacks reliably without giving false alarms. It is very important that the false alarm rate is low as in a live network with large amount of traffic, the number of false alarms may exceed the total number of attacks detected correctly thereby decreasing the confidence in the attack detection capability of the system. Ideally, the system must detect all intrusions with no false alarms. The challenge is to build a system which has broad attack detection coverage, i.e. it can detect a wide variety of attacks and at the same time which results in very few false alarms.

2. The system must be able to handle large amount of data without affecting performance and without dropping data, i.e. the rate at which the audit patterns are processed and decision is made must be greater than or equal to the rate of arrival of new audit patterns. Hence the speed of operation is critical for systems deployed in high speed networks. In addition, the system must be capable of operating in real-time by initiating a response mechanism once an attack is detected. The challenge is to prevent an attack rather than simply detecting it.

3. A system which can link an alert generated by the intrusion detector to the actual security incident is desirable. Such a system would help in quick analysis of the attack and may also provide effective response to intrusion as opposed to a system which offers no after attack analysis. Hence, it is not only necessary to detect an attack, but it is also important to identify the type of attack.

4. It is desirable to develop a system which is resistant to attacks since, a system that can be exploited during an attack may not be able to detect attacks reliably.

5. Every network and application is different. The challenge is to build a system which is scalable and which can be easily customized as per the specific requirements of the environment where it is deployed.

2.3 Classification of Intrusion Detection Systems

Classifying intrusion detection systems helps to better understand their capabilities and limitations. We therefore, present the classification of intrusion detection systems in Figure 2.1.

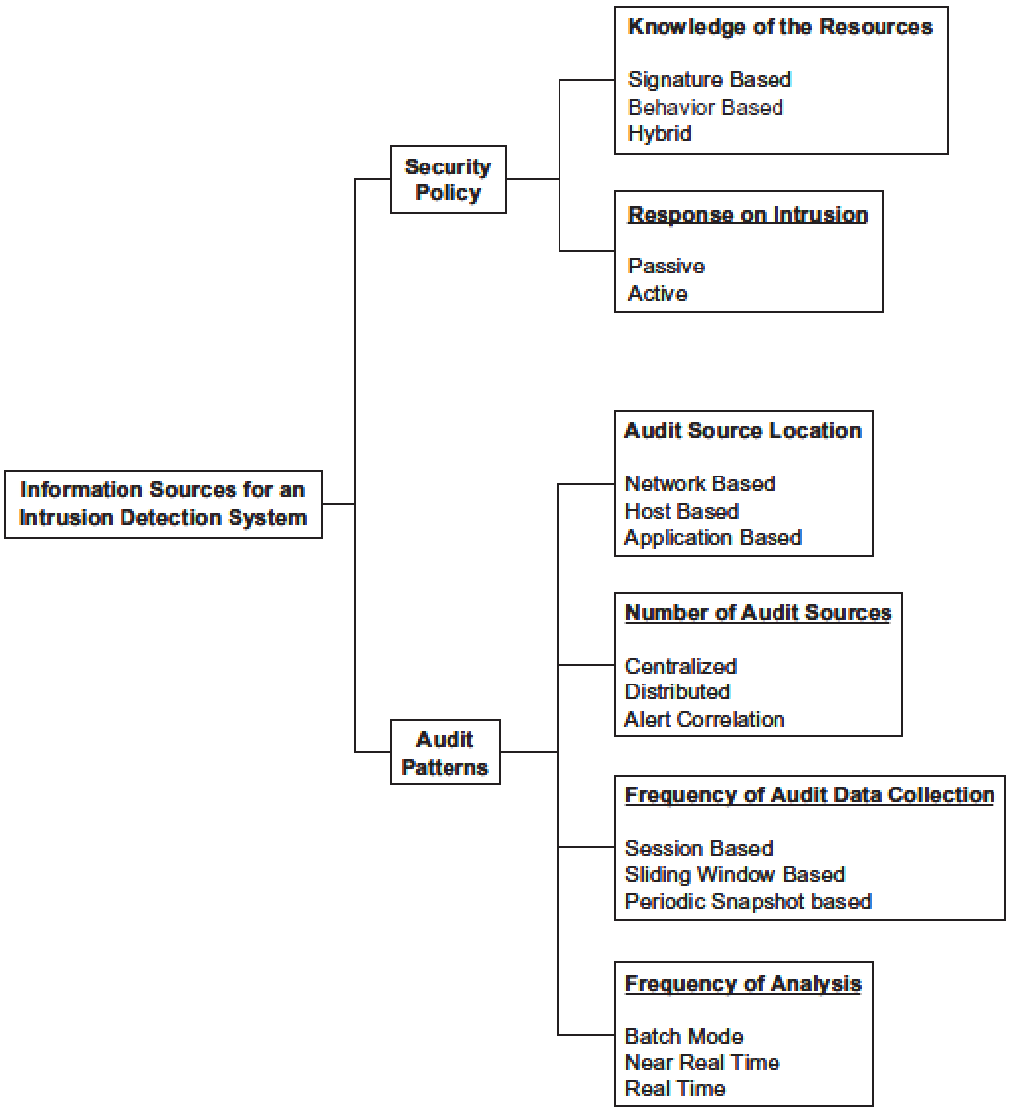


Figure 2.1 Classification of Intrusion Detection System

From Figure 2.1, we observe that for any intrusion detection system, security policy and audit patterns are the two prime information sources. The audit patterns must be analyzed to detect an attack and the security policy defines the acceptable and non acceptable usage of a resource and helps to qualify whether an event is *normal* or *attack*. Hence, based on the given classification, an example of an intrusion detection system can be a centralized system deployed on a network with sliding window based data collection which operates in real-time and is based on signature analysis with active response to intrusion.

2.3.1 Classification based upon the Security Policy definition

Intrusion detection systems are classified in two ways based upon the security policy definition.

1. Security policy defines the normal and abnormal usage of every resource. Consider a set *U*, which represents the complete domain (universe) for a resource *R*. The set *U* consists of both, *normal* and *abnormal* usage of *R*. Hence, *U* = *UR¡normal*, *UR¡attack*. The problem is to identify the set *U* such that it is complete and unambiguous. However, in most practical situations it is very difficult to identify and define the complete set *U* and only a small portion of this set is available which is denoted as *S*. Hence, the security policy is defined with only the knowledge contained in the subset *S*, where *S* = *SR¡normal*, *SR¡attack*. This is represented in Figure 2.2.

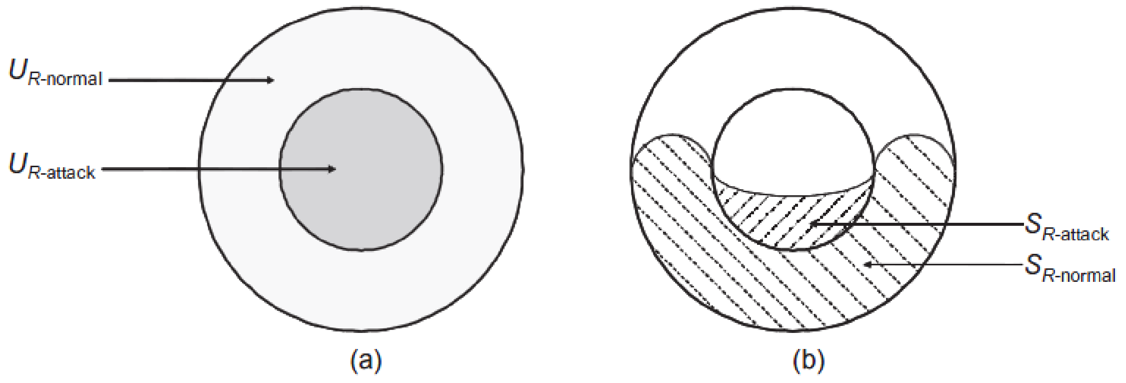


Figure 2.2 Representation for a resource R: (a) complete knowledge, and (b) available knowledge, where |UR−normal| ≥ |SR−normal| and |UR−attack| ≥|SR−attack|.

Based upon the elements of subset *S*, intrusion detection system can be classified as:

(a) Signature (Misuse) Based – When the set *S* only contains the events which are known to be *attack*, the system focuses on detecting known misuses and is known as signature or misuse based system [53]. Signature based system are represented in Figure 2.3.

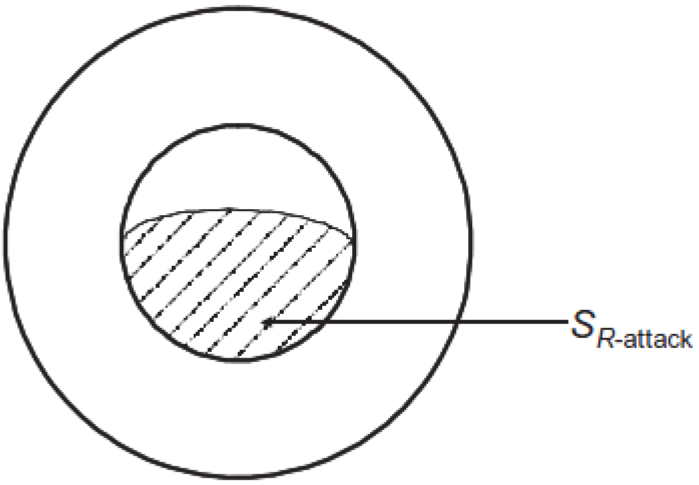


Fig. 2.3. Representation of a signature based system.

Signature based systems employ pattern matching approaches to detect attacks. They can detect attacks with very few false alarms but have limited attack detection capability since they cannot detect unseen attacks. Their attack detection capability is directly proportional to the available knowledge of attacks in the set *S*, i.e. knowledge of *SR¡attack*. To be effective, such systems require complete knowledge of attacks, i.e. *SR¡attack* should be equal to *UR¡attack*, which is not always possible

(b) Behavior (Anomaly) Based – When the set *S* only consists of events which are known to be *normal*, the goal of the intrusion detection system is to identify significant deviations from the known normal behavior [53] as shown in Figure 2.4.

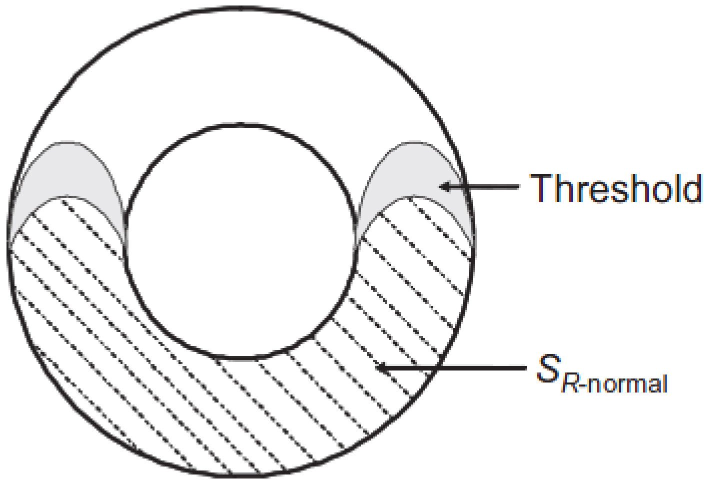


Figure2.4 Representation of a behavior based system.

For behavior based systems to be effective complete knowledge of normal behavior of a resource is required, i.e. the set *SR¡normal* should be equal to the set *UR¡normal*. Since the complete knowledge of a resource may not be available, a threshold is used which gives some flexibility to the system. Events which lie beyond the threshold are detected as attacks. Hence, behaviour based systems; in general, suffer from a large false alarm rate. False alarms can be reduced by increasing the threshold, however, this affects the attack detection and the system may not be able to detect a wide variety of attacks. Hence, there is a tradeoff in limiting the number of false alarms and the capability of the system to detect a variety of attacks.

(c) Hybrid – In most environments, it may not be possible to completely define either the normal or the abnormal behaviour. As a result, an intrusion detection system may generate a large number of false alarms or may be specific in detecting only a few types of attacks. A hybrid system uses the partial knowledge of both, i.e., *SR¡normal* and *SR¡attack*, to detect attacks; often resulting in fewer false alarms and detecting more attacks. Such systems generally employ machine learning approaches. A hybrid system is represented in Figure 2.5.

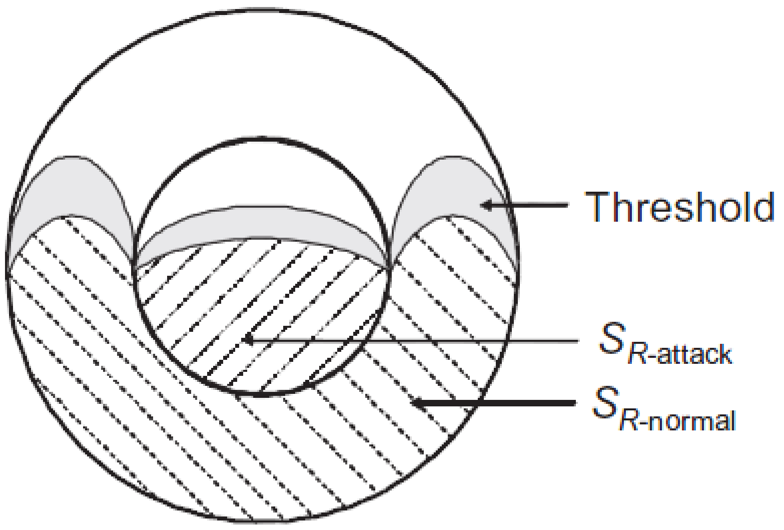


Figure 2.5: Representation of a Hybrid System

2. The **security policy** also defines how the system must respond when an attack is detected; based upon which the intrusion detection systems can be classified as:

(a) Passive Response Systems – In a passive response system, the system does not take any measure to respond to an attack once an attack is detected. It simply generates an alert which can be analyzed by the administrator at some later stage [15], [53].

(b) Active Response Systems – In active response systems, the intrusion detection systems respond to attacks by various possible approaches which may include blocking the source of the attack for a predefined time period [15], [53].

2.3.2 Classification based upon the Audit Patterns

1. The source from which the audit patterns are collected affects the attack detection capability

of a system. For example, when network statistics are used as the audit patterns, they cannot provide any detail about the user and system interaction. Based on this, intrusion detection systems are classified as:

(a) Network Based – In a network based system, the audit patterns collected at the network level are used by the intrusion detector [21], [22]. Though a single system (or a few strategically placed systems) is (are) sufficient for the entire network, the attack detection capability of a network based system is limited. This is because it is hard to infer the contextual information directly from the network audit patterns. Further, the audit patterns may be encrypted rendering them unusable by the intrusion detector at the network level. In addition, large amount of audit patterns at the network level may also affect the total attack detection accuracy. This is because of two reasons; first, a significant portion of the total incoming patterns may be allowed to pass into the network without any analysis and second, in high speed networks, it may be practical to analyze only the summary statistics collected at regular time intervals. These statistics may include features such as the total number of connections, amount of incoming and outgoing traffic. Such features only provide a high level summary which may not be able to detect attacks reliably [53].

(b) Host Based – The intrusion detector in a host based system analyzes the audit patterns generated at the kernel level of the system which include system access logs and the error logs [53]. The audit patterns collected at the individual host contains more specific information than the network level audit patterns, which may be used to detect attacks reliably. However, it becomes difficult to manage a large number f host based systems in a big network. Additionally, host based systems can themselves be the victims of an attack.

(c) Application Based – The application based systems are concerned only with a single application and detect attacks directed at a particular application or a privileged process [12]. They can analyze either the application access logs or the system calls generated by the processes to detect anomalous activities. The application based systems can be very effective as they can exploit the complete knowledge of the application and can be used even when encryption is used in communication. They can also analyze the user and application interactions which can significantly improve the attack detection accuracy.

2. In order to detect intrusions, the audit patterns can be collected from a single source or from a number of sources. When the audit patterns are collected from more than one source, the decision can be made by individual nodes or by aggregating the audit patterns at a single point and then analyzing them together. Based upon this property, the intrusion detection systems can be classified as:

(a) Centralized System – In a centralized system, the audit patterns are collected either from a single source or from multiple sources but are processed at a single point where they are analyzed together to determine the global state of the network [53]. However, such systems may themselves become a target of attacks.

(b) Distributed System – In contrast to the centralized systems, the distributed systems can make local decisions close to the source of the audit patterns and may report only a small summary of activities to a higher level in the system. The advantage of a distributed system for intrusion detection is that immediate response mechanism can be activated based upon local decisions. However, distributed systems can be less accurate due to lack of global knowledge. Agent based systems are examples of distributed intrusion detection systems [53].

(c) Alert Correlation – Alert correlation based systems analyze the alert generated by a number of cooperating intrusion detection system [15]. The individual systems may themselves be centralized or decentralized. Alert correlation systems can only be effective when multiple networks are attacked with similar attacks such as in case of worm outbreak. Incase when the attacks are network specific, the alert correlation systems will not be effective even though a few target networks may detect some anomalous activities. In such cases, the local alerts will be discarded as false alarms due to lack of global consensus.

3. Regardless of the source and the number of audit patterns, the intrusion detection systems

can be classified depending upon the frequency at which the audit patterns are collected. Based on this, they are classified as:

(a) Session Based – Audit patterns can be collected at the end of every session by summarizing different features. Methods can be used which analyze the summary of every session once the session is terminated.

(b) Sliding Window Based – In case of sliding window based collection of audit patterns, events are recorded using a moving window of fixed or variable width. The width of the window defines the number of events recorded together and the step size for sliding the window determines how fast the window is advanced forward.

(c) Periodic Snapshot Based – Instead of recording every event or summarizing a session at its termination, snapshots of different states of the entire system can be taken at regular intervals which can be analyzed to detect intrusions.

4. Depending upon the frequency of analysis of audit patterns, the intrusion detection systems can be classified as:

(a) Batch Mode – In batch mode intrusion detection, the audit patterns are aggregated in a central repository. The patterns are then analyzed for intrusions at predefined time intervals. Such systems cannot provide any immediate response to intrusion and can only perform the recovery task once an attack is detected.

(b) Near Real-time – An intrusion detection system is said to perform in near real-time when the system cannot detect an intrusion when it commenced, but can detect it at some later stage during the attack or immediately at the end of an attack. In such systems, there is some delay before the patterns are made available to the intrusion detector. Patterns collected by taking periodic snapshots or using moving window with step size greater than one can be used for near real-time intrusion detection.

(c) Real-time – A real-time intrusion detection system must detect an attack as soon as it is commenced, i.e. the system is said to perform in real-time if and only if, for an event ‘x’ when the attack commenced, the attacker cannot succeed with the event ‘x+1’. Hence, for real-time intrusion detection, the system must detect an attack immediately. However, in practice it is very difficult to build such a system given the constraint that it should have low false alarm rate and high attack detection accuracy. Real-time intrusion detection systems can be implemented by using a moving window with a step of size one. Network based signature detection systems, which perform pattern matching can also perform in real-time by checking every event for known attacks. However, they are limited in detecting only those attacks whose signatures are known in prior. A typical example is the Snort [23].

2.4 Audit Patterns

The raw patterns must be preprocessed and presented in a format which can be interpreted by the intrusion detector before they can be analyzed.

2.4.1 Properties of Audit Patterns useful for Intrusion Detection

Different properties in the audit patterns can be analyzed for detecting intrusions. The authors has described in [49] three properties which can be used to detect intrusions.

1. Frequency of Event(s) – Frequency determines how often an event occurs in a predefined time interval. A threshold can be used to define the limit. When the frequency crosses this limit, an alarm can be raised. Properties such as the number of invalid login attempts and the number of rows accessed in a database can be used to measure frequency.

2. Duration of Event(s) – Rather than counting the number of occurrences of an event, the duration property determines the acceptable time duration for a particular event. It is based upon selecting a threshold which defines an acceptable range for a particular event. For example, large number of invalid login attempts for a single user id in a very short time span can be considered as an attempt to guess a password and hence an attack. Systems analyzing the frequency or (and) duration property for the events can perform efficiently but they suffer from large false alarm rate as it is often difficult to determine the correct threshold for the events.

3. Ordering of Events – Analyzing the order in which events occur can improve the attack detection accuracy and reduce false alarms. This is because, very often, intrusion is a multi step process in which a number of events must occur sequentially in order to launch a successful attack. However, to avoid detection from systems which do analyze a sequence of events, the attacks can be spread over a long period of time such that the events cannot be correlated unless a long history is maintained by the intrusion detection system. A system which can analyze all of the above mentioned properties can detect attacks with high accuracy. However, such a system may be inefficient in operation.

2.4.2 Univariate or Multivariate Audit Patterns

The audit patterns used to detect attacks may either be univariate or multivariate. As, discussed before, the audit patterns may be collected from the routers and switches for the network level systems. When only one feature is analyzed, in case of univariate audit patterns, the analysis is much simpler in comparison to when many features are analyzed together, as in case of multivariate analysis. However, a single feature itself may not be the complete representation and, hence, insufficient to detect attacks. For example, when the sequence of system calls generated by a privileged process is analyzed for detecting abnormal behavior, discarding other features such as the parameters of the system calls can affect the attack detection capability of the system [24].

2.4.3 Relational or Sequential Representation

Very often, the audit patterns collected are sequential where one or more features are recorded continuously. However, the raw audit patterns may be processed into a relational form and a number of new features can be added. These features often give a high level representation of audit patterns in a summarized form. Examples of such features include; total amount of data transferred in a session and duration of a session. Frequency and duration properties of events can be easily represented in a relational form. Converting the audit patterns from sequential to relational form has two advantages; first, more features can be added and second, efficient methods can be used for analysis of audit patterns in relational form. However, this may result in affecting the attack detection capability as in relational form the ordering of events and, hence, the relationship among sequential events is lost. When the audit patterns are represented sequentially, event ordering can be exploited in favour of higher attack detection accuracy. However, in general, sequence analysis is slower when compared to the relational analysis.

2.5 Evaluation Metrics

Evaluating different methods for detecting intrusions is important. Intrusion detection is an example of a problem with imbalanced classes, i.e. the number of instances in the classes is not equally distributed. The number of attacks is very small when compared with the total number of normal events. Note that, in case of the Denial of Service attacks, the amount of attack traffic is extremely large as compared to the normal traffic. Hence, evaluating intrusion detection systems using simple accuracy metric may result in very high accuracy [25]. Other metrics such as *Precision*, *Recall* and *F-Measure*, which do not depend on the size of the test set, are, thus, used for evaluating intrusion detectors. These are defined with the help of the confusion matrix as follows:

	Predicted Normal	*Predicted Attack*
True Normal	True Negative	False Positive
True Attack	False Negative	True Positive

Table 2.1: Confusion Matrix

Where *b* corresponds to the relative importance of Precision vs. Recall and is usually set to 1. Hence, a system must have high Precision (i.e. it must detect only attacks), high Recall (i.e. it must detect all attacks) and, thus, a high F-Measure.

In addition to evaluating the attack detection capability of the detector, time taken to detect an attack is also significant. The time performance is generally measured for the time taken by the intrusion detector to detect an attack from the time the audit patterns are fed into the detector.

This is sufficient for comparison when different methods use exactly the same data for analysis, however, it does not represent the efficiency of the intrusion detection system, since the time taken in collecting and preprocessing the audit patterns is not considered. Hence, in real environments, total time must be measured which is the time from the point when intrusion actually started to the point in time when the response mechanism is activated.

2.6 Literature Review

Two most significant motives to launch attacks as described in [82] are, either to force a network to stop some service(s) that it is providing or to steal some information stored in a network. An intrusion detection system must be able to detect such anomalous activities. However, what is normal and what is anomalous is not defined, i.e., an event may be considered normal with respect to some criteria, but the same may be labeled anomalous when this criterion is changed. Hence, the objective is to find anomalous test patterns which are similar to the anomalous patterns which occurred during training. The underlying assumption is that the evaluating criterion is unchanged and the system is properly trained such that it can reliably separate normal and anomalous events.

2.6.1 Frameworks for building Intrusion Detection Systems

A number of frameworks have been proposed for building intrusion detection systems. The common intrusion detection framework is described in [20]. The authors in [25] and [26] describe a data mining framework for building intrusion detection systems. Using the approach described in [26], the rules can be learned inductively instead of manually coding the intrusion patterns and profiles. However, their approach requires the use of a large amount of noise free audit data to train the models. Agent based intrusion detection frameworks are discussed in [27] and [28]. Frameworks which describe the collaborative use of intrusion detection systems have also been proposed [29], [30]. The system described in [29] is based on the combination of network based and host based systems while the system in [30] employs both, signature based and behavior based techniques for detecting intrusions. All of these frameworks suffer from one major drawback; a single intrusion detector used within these frameworks is trained to detect a wide variety of attacks. This results in a large number of false alarms.

2.6.2 Network Intrusion Detection

The prospect of maintaining a single system which can be used to detect network wide attacks make network monitoring a preferred option as opposed to monitoring individual hosts in a large network. A number of techniques such as association rules, clustering, naive Bayes classifier, support vector machines, genetic algorithms, artificial neural networks and others have been applied to detect intrusions at network level. It is important to note that different methods are based on specific assumptions and analyze different properties in the audit patterns, resulting in different attack detection capabilities. These methods can be broadly divided into three major categories:

Pattern Matching

Pattern matching techniques search for predefined set of patterns (known as signatures) in the audit patterns to detect intrusions. Pattern matching approaches are employed on the audit patterns which do not have any state or sequence information. Hence, they assume independence among events. However, this assumption may not always hold as a single intrusion may span over multiple events which are correlated. The prime advantage of pattern matching approaches is that they are very efficient and triggers an alert only when an exact match of an attack signature is found resulting in very few false alarms. They can, however, detect attacks only if the corresponding pattern (signature) exists in the signature database. Hence, they cannot detect unseen attacks, for which there are no signatures [24], [53]. Snort system [23] is based upon pattern matching.

Statistical Methods

Statistical methods based on modeling the monitored variables as independent Gaussian random variables and methods such as those based on the Hotelling *T*2 test statistic can be used to detect attacks by calculating deviations in the present profile from the stored normal profile [24]. They are based upon modeling the underlying process which generates the audit patterns and exploit the frequency and duration property of events. They often analyze properties such as the overall system load and statistical distribution of events, which represent a summary measure. When the deviations exceed a predefined threshold, the system triggers an alarm. To determine this threshold accurately is a critical issue. When the threshold is low, the system raises a large number of (false) alarms and when the threshold is high, the system may not detect attacks reliably. Though these methods can handle multiple features in the audit patterns, very often, in order to reduce complexity and improve system performance only a single feature is considered, as in the Intrusion Detection Expert System (IDES) [54].This, however, affects the attack detection accuracy. Statistical methods can operate either in batch mode (Haystack system) or in real-time mode (IDES).

Data Mining and Machine Learning

Data mining and machine learning methods focus on analyzing the properties of the audit patterns rather than identifying the process which generated them [24]. These methods include approaches for mining association rules, classification and cluster analysis. Classification methods are one of the most researched and include methods like the decision trees, Bayesian classifiers, artificial neural networks, k-nearest neighbor classification, support vector machines and many others.

Clustering – Clustering of data has been applied extensively for intrusion detection using a number of methods such as k-means, fuzzy c-means and others [31], [32]. Clustering methods are based upon calculating the numeric distance of a test point from different cluster centers and then adding the point to the closest cluster. One of the main drawbacks of clustering technique is that since a numeric distance measure is used, the observations must be numeric. Observations with symbolic features cannot be readily used for clustering which results in inaccuracy. In addition, clustering methods consider the features independently and are unable to capture the relationship between different features of a single record which results in lower accuracy. Another issue when applying any clustering method is to select the distance measure as different distance measures result in clusters with different shapes and sizes. Frequently used distance measures are the Euclidian distance and the Mahalanobis distance [24]. Clustering can, however, be performed in case only the normal audit patterns are available. In such cases, density based clustering methods can be used which are based on the assumption that intrusions are rare and dissimilar to the normal events. This is similar to identifying the outlier points which can be considered as intrusions.

Data Mining – Data mining approaches [25], [26] are based on mining association rules[33] and using frequent episodes [34] to build classifiers by discovering relevant patterns of program and user behaviour. Association rules and frequent episodes are used to learn the record patterns that describe user behaviour. These approaches can deal with symbolic features and the features can be defined in the form of packet and connection details. Mining association rules for intrusion detection has the advantage that they are easy to interpret. However, they are based upon building a database of rules of normal and frequent items during the training phase. During testing, patterns from the test data are extracted and various classification methods can be used to classify the test data. The detection accuracy suffers as the database of rules is not a complete representation of the normal audit patterns.

Bayesian Classifiers – Naive Bayes classifiers are also proposed for intrusion detection [35]. However, they make strict independence assumption between the features in an observation resulting in lower attack detection accuracy when the features are correlated, which is often the case. Bayesian network [36] can also be used for intrusion detection [37], [38]. However, they tend to be attack specific and build a decision network based on special characteristics of individual attacks. As a result, the size of a Bayesian network increases rapidly as the number of features and the type of attacks modeled by the network increases.

Decision Trees – Decision trees have also been used for intrusion detection [35], [39]. Decision trees select the best features for each decision node during tree construction based on some well defined criteria. One such criterion is the gain ratio which is used in C4.5. Decision trees generally have very high speed of operation and high attack detection accuracy and have been successfully used to build effective intrusion detection systems.

Artificial Neural Networks – Neural networks have been used extensively to build network intrusion detection systems. Though, the neural networks can work effectively with noisy data, like other methods, they require large amount of data for training and it is often hard to select the best possible architecture for the neural network.

Support Vector Machines – Support vector machines map real valued input feature vector to higher dimensional feature space through nonlinear mapping and have been used for detecting intrusions [40], [41], [42]. They can provide real-time attack detection capability, deal with large dimensionality of data and perform multi class classification.

For data mining and machine learning based approaches, the accuracy of the trained system also depends upon the amount of audit patterns available during training. Generally, training with more audit patterns result in a better model. The above discussed methods often deal with the summarized representation of the audit patterns and may analyze multiple features which are considered independently. The prime reason for working with summary patterns is that the system tends to be simple, efficient and give fairly good attack detection accuracy. Similar to the pattern matching and statistical methods, these methods assume independence among consecutive events and hence do not consider the order of occurrence of events for attack detection.

Markov Models – Markov chains [43], [44] and hidden Markov models [45] can be used when dealing with sequential representation of audit patterns. [12], [46], [47] and [48] describes the use of hidden Markov model for intrusion detection. Hidden Markov models have been shown to be effective in modeling sequences of system calls of a privileged process, which can be used to detect anomalous traces. However, modeling system calls alone may not always provide accurate classification as various connection level features are ignored.

Further, hidden Markov models cannot model long range dependencies between the observations [84]. Very often the sequence itself is a vector and has many correlated features. However, in order to gain computational efficiency the multivariate data analysis problem is broken into multiple univariate data analysis problems and the individual results are combined using a voting mechanism [24]. This however, results in inaccuracy as the correlation among the features is lost. The authors in [84] show that modeling the ordering property of events, in addition to the duration and frequency, results in higher attack detection accuracy. The drawback with modeling the ordering of events is that the complexity of the system increases which affects the performance of the system. Hence, there is a tradeoff between detection accuracy and the time required for attack detection.

A number of intrusion detection systems such as the IDES (Intrusion Detection Expert System), Haystack system, the MIDAS (Multics Intrusion Detection System), W&S (Wisdom and Sense) system, TIM (Time based Inductive Machine), Snort and others have been developed which operate at the network level [1]. However, network intrusion detection systems must perform very efficiently in order to handle large amount of network data and hence many of the network intrusion detection systems are primarily based on signature matching. When anomaly detection systems are used at network level, they either consider only one feature [54] or assume the features to be independent [85]. However, we propose to use a hybrid system based on conditional random fields and integrate the layered framework to build a single system which can operate in high speed networks and can detect a wide variety of attacks with very few false alarms.

2.7 Layered approach for intrusion detection

We now describe the Layer-based Intrusion Detection System (LIDS) in detail. The LIDS draws its motivation from what we call as the Airport Security model, where a number of security checks are performed one after the other in a sequence. Similar to this model, the LIDS represents a sequential Layered Approach and is based on ensuring availability, confidentiality, and integrity of data and (or) services over a network. Fig. 2.7 gives a generic representation of the framework.

The goal of using a layered model is to reduce computation and the overall time required to detect anomalous events. The time required to detect an intrusive event is significant and can be reduced by eliminating the communication overhead among different layers. This can be achieved by making the layers autonomous and self-sufficient to block an attack without the need of a central decision-maker. Every layer in the LIDS framework is trained separately and then deployed sequentially. We define four layers that correspond to the four attack groups mentioned in the data set. They are Probe layer, DoS layer, R2L layer, and U2R layer. Each layer is then separately trained with a small set of relevant features.

Feature selection is significant for Layered Approach and discussed in the next section. In order to make the layers independent, some features may be present in more than one layer. The layers essentially act as filters that block any anomalous connection, thereby eliminating the need of further processing at subsequent layers enabling quick response to intrusion. The effect of such a sequence of layers is that the anomalous events are identified and blocked as soon as they are detected. Our second goal is to improve the speed of operation of the system. Hence, we implement the LIDS and select a small set of features for every layer rather than using all the 41 features. This results in significant performance improvement during both the training and the testing of the system. In many situations, there is a trade-off between efficiency and accuracy of the system and there can be various avenues to improve system performance. Methods such as naive Bayes assume independence among the observed data. This certainly increases system efficiency, but it may severely affect the accuracy. To balance this trade-off, we use the CRFs that are more accurate, though expensive, but we implement the Layered Approach to improve overall system performance. The performance of, layered CRFs, is comparable to that of the decision trees and the naive Bayes, and our system has higher attack detection accuracy.

Advantages of Layered Framework

We now summarize the advantages of using our layered framework. Using our layered framework improves attack detection accuracy and the system can detect a wide variety of attacks by making use of the domain specific knowledge. The layered framework does not degrade system performance as individual layers are independent and are trained with only a small number of features, thereby, resulting in an efficient system. Additionally, using our layered framework opens avenues to perform pipelining resulting in very high speed of operation. Implementing pipelining, particularly in multi core processors, can significantly improve the performance by reducing the multiple I/O operations to a single I/O operation since all the features can be read in a single operation and analyzed by different layers in the layered framework. Our framework is easily customizable and the number of layers can be adjusted depending upon the requirements of the target network.

Layered framework is not restrictive in using a single method to detect attacks. Different methods can be seamlessly integrated in our framework to build effective intrusion detectors. Layered framework for building effective and efficient network intrusion detection systems fits well in the traditional layered defense approach for providing network and enterprise level security.

Layered framework has the advantage that the type of attack can be inferred directly from the layer at which it is detected. As a result, specific intrusion response mechanisms can be activated for different attacks.

Comparison with other Frameworks

Ensuring continuity of services and security of data from unauthorized disclosure and malicious modifications are critical for any organization. However, providing a desired level of security at the enterprise level can be challenging. No single tool can provide enterprise wide security and hence, a number of different security tools are deployed. For this, a layered defense approach is often employed to provide security at the organizational level. This traditional layered defense approach incorporates a variety of security tools such as the network surveillance, perimeter access control, firewalls, network, host and application intrusion detection systems, file integrity checkers, data encryption and others which are deployed at different access points in a layered security framework. The traditional layered architecture is perceived as a framework for ensuring complete organizational security rather than as an approach for building effective and efficient intrusion detection systems. Figure 3.2 represents the traditional layered defense approach. However, as discussed earlier, we present a layered framework for building intrusion detection systems. Our framework fits well in the traditional layered defense approach and can be used to develop effective and efficient network intrusion detection systems. Further, the four components viz., event generators, event Analyzers, event Databases and the response units, presented in the Common Intrusion Detection Framework [20] can be defined for every intrusion detection sub system in our layered framework.

In the data mining framework for intrusion detection, [50], the authors describe the use of data mining algorithms to compute activity patterns from system audit data to extract features which are then used to generate rules to detect intrusions. The same approach can be applied for building an intrusion detection system based on our layered framework. Our framework can not only seamlessly integrate the use of data mining technique for intrusion detection, but can Figure 3.2: Traditional Layered Defense Approach to Provide Enterprise Wide Security also help to improve its performance by selecting only a small number of significant features for building separate intrusion detection sub systems which can be used to effectively detect different classes of attacks at different layers.

2.8 Conditional Random Fields

Conditional models are probabilistic systems which are used to model the conditional distribution over a set of random variables. Such models have been extensively used in natural language processing tasks and computational biology. Conditional models offer a better framework as they o not make any unwarranted assumptions on the observations and can be used to model rich overlapping features among the visible observations. Maxent classifiers [55], [56], [57] maximum entropy Markov models [66], [58] and conditional random fields [49] are such conditional models. The simplest conditional classifier is the Maxent classifier based upon maximum entropy classification which estimates the conditional distribution of every class given the observations. The training data is used to constrain this conditional distribution while ensuring maximum entropy and hence maximum uniformity. We now give a brief description of the conditional random fields which is motivated from the work in [49]. Let X be the random variable over a data sequence to be labeled and Y be the corresponding label sequence. Also, let $G = (V, E)$ be a graph such that $Y = (Y_v)$ so that Y is indexed by the vertices of G. Then (X,Y) is a conditional random field, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: : $p(Y_v|X,\ Y_w,\ w \neq v) = p(Y_v|X,\ Y_w,\ w \sim v)$where $w \sim v$ means that w and v are neighbors in G, i.e. a conditional random field is a random field globally conditioned on X. For a simple sequence (or chain) modeling, as in our case, the joint distribution over the label sequence Y given X has the form:

$$p_\theta(y|x) \propto \exp(\textstyle\sum_{e\in E,k} \lambda k f k(e, y|e, x) + \sum_{v\in V,k} \mu k g k(v, y|v, x))$$

where x is the data sequence, y is a label sequence, and yjs is the set of components of y associated with the vertices or edges in sub graph S. The features fk and gk are assumed to be given and fixed.

Further, the parameter estimation problem is to find the parameters $q = (l_1, l_2, ...; m_1, m_2, ...)$ from the training data $D = (x^i, y^i)_{i=1}{}^N$ with the empirical distribution $\tilde{}\,p(x, y)$.

The graphical structure of a conditional random field is represented in Figure 2.6 where x_1, x_2, x_3, x_4 represents an observed sequence of length four and every event in the sequence are correspondingly labeled as y_1, y_2, y_3, y_4.

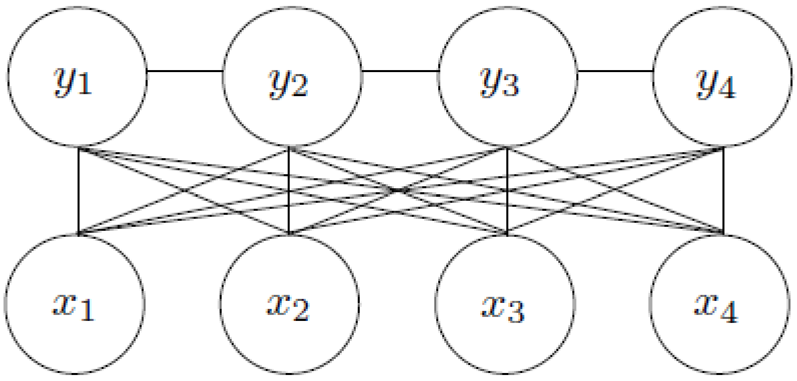


Figure 2.6 Graphical Representation of a Conditional Random Field

The prime advantage of conditional random fields is that they are discriminative models which directly model the conditional distribution *p*(*y/x*). Further, conditional random fields are undirected models and free from label bias and observation bias which are present in other conditional models [59]. Generative models such as the Markov chains, hidden Markov models and joint distribution have two disadvantages. First, the joint distribution is not required since the observations are completely visible and the interest is in finding the correct class which is the conditional distribution *p*(*y/x*). Second, inferring conditional probability *p*(*y/x*) from the joint distribution, using the Bayes rule, requires marginal distribution *p*(*x*) which is difficult to estimate as the amount of training data is limited and the observation *x* contains highly dependent features. As a result strong independence assumptions are made to reduce complexity. This results in reduced accuracy [60]. Instead, conditional random fields predict the label sequence *y* given the observation sequence *x*, allowing them to model arbitrary relationships among different features in the observations without making independence assumptions.

Conditional random fields, thus, offer us the required framework to build effective intrusion detection systems. The task of intrusion detection can be compared to many problems in machine learning, natural language processing and bio-informatics such as gene prediction, determining secondary structures of protein sequences, part of speech tagging, text segmentation, shallow parsing, named entity recognition, object recognition and many others. The conditional random fields have proven to be very successful in such tasks. Hence, in this thesis, we explore the suitability of conditional random fields for building robust intrusion detection systems.

2.9 FCANN

Artificial Neural Network (ANN) is one of the widely used techniques and has been successful in solving many complex practical problems. And ANN has been successfully applied into IDS [76] [71]. However, the main drawbacks of ANN-based IDS exist in two aspects:

(1) Lower detection precision, especially for low-frequent attacks, e.g., Remote to Local (R2L), User to Root (U2R), and
(2) Weaker detection stability [77].

For the above two aspects, the main reason is that the distribution of different types of attacks is imbalanced. For low-frequent attacks, the leaning sample size is too small compared to high-frequent attacks. It makes ANN not easy to learn the characters of these attacks and therefore detection precision is much lower. In practice, low-frequent attacks do not mean they are unimportant. Instead, serious consequence will be caused if these attacks succeeded. For example, if the U2R attacks succeeded, the attacker can get the authority of root user and do everything he likes to the targeted computer systems or network device. Furthermore in IDS the low-frequent attacks are often outliers. Thus ANN is unstable as it often converges to the local minimum [69]. Although prior research has proposed some approaches, when encountering large datasets, these approaches become not effective [75] [24]. To solve the above two problems, we propose a novel approach for ANN-based IDS, FC-ANN, to enhance the detection precision for low-frequent attacks and detection stability. The general procedure of FC-ANN approach has the following three stages. In the first stage, a fuzzy clustering technique is used to generate different training subsets. Based on different training sets, different ANNs are trained in the second stage. In the third stage, in order to eliminate the errors of different ANNs, a meta-learner, fuzzy aggregation module, is introduced to learn again and combine the different ANN's results. The whole approach reflects the famous philosophy ‘‘divide and conquer”. By fuzzy clustering, the whole training set is divided into subsets which have less number and lower complexity. Thus the ANN can learn each subset more quickly, robustly and precisely, especially for low-frequent attacks, such as U2R and R2L attacks. To illustrate the applicability and capability of the new approach, the results of experiments on KDD CUP 1999 dataset.

Chapter 3
Layered Framework for Building Intrusion Detection Systems

The LIDS draws its motivation from what we call as the Airport Security model, where a number of security checks are performed one after the other in a sequence. Today's networks and enterprises use Layered Defense Approach to ensure security at different access levels. A variety of tools such as network surveillance, perimeter access control, firewalls network, host and application intrusion detection systems, data encryption are used at different layers to detect attacks at that layer. However, with the rapid increase in the number and type of attacks, a single system is not effective enough given the constraints of achieving high attack detection accuracy and high system throughput. Hence, a layered framework for building intrusion detection systems which can be used, for example, to build a network intrusion detection system which can detect a wide variety of attacks reliably and efficiently when compared to the traditional network intrusion detection systems. Another advantage of our Layer based Intrusion Detection System (LIDS) framework is that it is very general and easily customizable depending upon the specific requirements of individual networks.

3.1 Introduction

The characteristics required for the intrusion detection system are as follows:

1. Broad attack detection coverage i.e., an intrusion detection system must detect different type of attacks effectively
2. Efficiency in operation, i.e. it must operate efficiently in high traffic networks

Today's networks are susceptive to a number of attacks, a large number of which are previously known. However, the number of previously unseen attacks is on a rise [3]. Signature based systems using pattern matching approaches can be used effectively and efficiently to detect previously known attacks in high speed networks. However, even a slight variation in attacks may not be detected by a signature based system. As a result, anomaly and hybrid systems are used to detect previously unseen attacks and have been proven to be more reliable in detecting novel attacks when compared with the signature based systems. A common practice to build anomaly and hybrid intrusion detection systems is to train a single system with labeled data to build a classifier which can then be used to detect attacks from a previously unseen test set.

At times, when labeled data is not available, clustering based systems can be used to distinguish between legitimate and malicious packets. However, a significant disadvantage of such systems is that they result in a large number of false alarms. The attack detection coverage of the system is further affected when a single system is trained to detect different type of attacks. To maximize attack detection, various systems such as [30] and [61] employ both the signature based and the anomaly based systems together. However, the anomaly based systems still remain a bottleneck in the joint system. This is because, a single anomaly detector is trained which is expected to accurately detect a variety of attacks and perform efficiently.

Thus, for a network intrusion detection system monitoring the incoming and outgoing network traffic and ensuring confidentiality, integrity and availability via a single system may not be possible due to several reasons including the complexity and the diverse type of attacks at the network level. Ensuring high speed of operation further limits the deployment, particularly, of anomaly and hybrid network intrusion detection systems. Network monitoring using a network intrusion detection system is only a single line of defense in the traditional layered defense approach which aims to provide complete organizational security. Hence, network intrusion detection systems are complemented by a variety of other tools such as network surveillance, perimeter access control, firewalls, host and application intrusion detection systems, file integrity checkers, data encryption and others and are deployed at different access points in a layered organizational security framework [62]. In this chapter we propose a layered framework for building anomaly and hybrid network intrusion detection systems which can operate efficiently in high speed networks and can accurately detect a variety of attacks. Layered framework is very general and can be easily customized by adding domain specific knowledge as per the specific requirements of the network in concern, thereby, giving flexibility in implementation.
The rest of the chapter is organized as follows; we give motivating examples to highlight the significance of the layered framework for intrusion detection in Section 3.2. We then describe

our layered framework in Section 3.3. We highlight the advantages of our framework in Section 3.4 and compare the layered framework with others in Section 3.5.

3.2 Motivating Examples

Anomaly and hybrid intrusion detection systems typically employ various data mining and machine learning based approaches which are inefficient when compared to the signature based systems which employ pattern matching. Hence, it becomes critical to search for methods which can be used to build efficient anomaly and hybrid intrusion detection systems. However, given that the present networks are prone to a wide variety of attacks, using a single system would not only degrade performance but will also be less effective in attack detection.

Consider for example, a single network intrusion detection system which is deployed to detect every network attack in a high speed network. A network is prone to different types of attacks such as the *Denial of Service (DoS)*, *Probe* and others. We note that the *DoS* and *Probe* attacks are different and require different features for their effective detection. When same features are used to detect the two attacks the accuracy decreases. It also makes the system bulky which affects its speed of operation. Hence, for effective attack detection, a network intrusion detection system must differentiate between different types of attacks. Thus, using a single system is not a viable option. One possible solution is having a number of sub systems each of which is specific in detecting a single category of attack (such as *DoS*, *Probe* and others). This is not only more effective in detecting individual classes of attacks, but it also results in an efficient system. The number of sub systems to be used can be determined by analyzing the potential risks and the availability of resources at individual installations. Hence, we propose a layered framework for building efficient anomaly and hybrid intrusion detection systems where different layers in the system are trained independently to detect different type of attacks with high accuracy. For example, based on Layered framework a network intrusion detection system may consist of four layers, where the layers correspond to four different attack classes; *Denial of Service*, *Probe*, *Remote to Local* and *User to Root*.

3.3 Description of our Framework

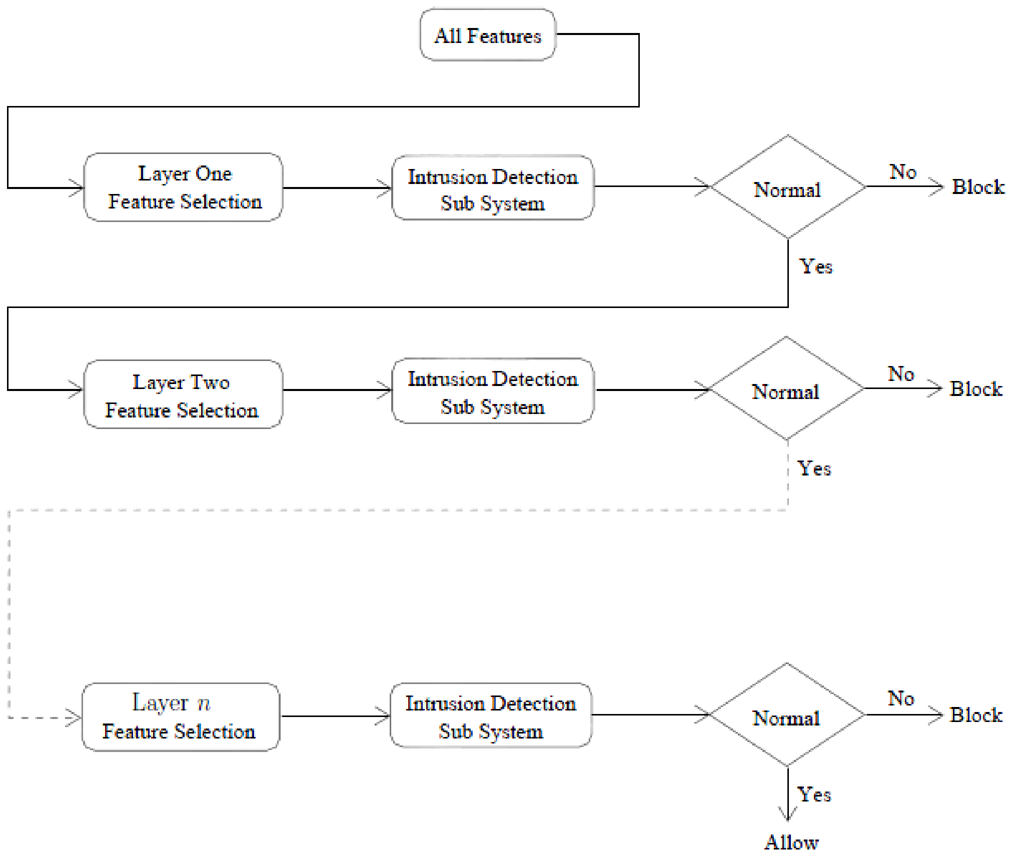

All Features
Layer One Feature Selection
Intrusion Detection Sub System
Normal
No
Block
Yes
Layer Two Feature Selection
Intrusion Detection Sub System
Normal
No
Block
Yes
Layer n Feature Selection
Intrusion Detection Sub System
Normal
No
Block
Yes
Allow

Figure 3.1 Framework for building LIDS

Figure 3.1 represents our framework for building Layer based Intrusion Detection Systems (LIDS). The figure represents an '*n*' layer system where every layer in itself is a small intrusion detection system which is specifically trained to detect only a single type of attack, for example the *DoS* attack. A number of such sub systems are then deployed sequentially, one after the other. This serves dual purpose; first, every layer can be trained with only a small number of features which are significant in detecting a particular class of attack. Second, the size of the sub system remains small and hence, it performs efficiently. A common disadvantage of using a modular approach, similar to our layered framework, is that it increases the communication overhead among the modules (sub systems). However, this can be easily eliminated in our framework by making every layer completely independent of every other layer. As a result, some features may be present in more than one layer. Depending upon the security policy of the network, every layer can simply block an attack once it is detected without the need of a central decision maker. A number of such layers essentially act as filters, which blocks anomalous connection as soon as they are detected in a particular layer, thereby providing a quick response to intrusion and simultaneously reducing the analysis at subsequent layers. It is important to note that a different response may be initiated at different layers depending upon the class of attack the layer is trained to detect. The amount of audit data analyzed by the system is more at the first layer and decreases at subsequent layers as more and more attacks are detected and blocked. In the worst case, when no attacks are detected until at the last layer, all the layers have the same load. However, the overall load for the average case is expected to be much less since attacks are detected and blocked at every subsequent layer. On the contrary, if the layers are arranged in parallel rather than in a sequence, the load at every sub system is same and is equal to that of the worst case in the sequential configuration. Additionally, the initial layers in the sequential configuration can be replicated to perform load balancing in order to improve performance.

3.3.1 Components of Individual Layers

Given that a network is prone to a wide variety of attacks, it is often not feasible to add a separate layer to detect every single attack. However, a number of similar attacks can be grouped together and represented as a single attack class. Every layer in our framework corresponds to a sub system which is trained independently to detect attacks belonging to a single attack class. As a result, the total number of layers in our framework remains small. For example, both, 'Smurf' and 'Neptune' result in *Denial of Service* and, hence, can be detected at a single layer rather than at two different layers.

Additionally, the layered framework is very general and the number of layers in the overall system can be adjusted depending upon the individual requirements of the network in concern.
Consider for example, a data repository which is a replica of a real-time application data and which does not provide any online services. To ensure security of this data, the priority is to simply detect network scans as opposed to detecting malicious data accesses. For such an environment, only a single layer which can reliably detect the *Probe* attacks is sufficient. Hence,

the number of layers in our framework can be easily customized depending upon the identified threats and the availability of resources.

Even though the number of layers and the significance of every layer in our framework depend upon the target network, every layer has two significant components:

1. Feature Selection Component – In order to detect intrusions, a large number of features can be monitored. These features include ‘protocol’, ‘type of service’, ‘number of bytes from source to destination’, ‘number of bytes from destination to source’, ‘whether or not a user is logged in’, ‘number of root accesses’, ‘number of files accessed’ and many others. However, to detect a single attack class, only a small set of these features is required at every layer. Using more features than required makes the system inefficient. For example, to detect *Probe* attacks, features such as the ‘protocol’ and ‘type of service’ are significant while features such as ‘number of root accesses’ and ‘number of files accessed’ are not significant.

2. Intrusion Detection and Response Sub System – The second component in every layer is the intrusion detection and response unit. To detect intrusions, our framework is not restrictive in using a particular anomaly or hybrid detector. A variety of previously well known intrusion detection methods such as the naive Bayes classifier, decision trees, support vector machines and others can be used. A prime advantage of our framework is that newer methods, such as conditional random fields as we will discuss in the following chapters, which are more effective in detecting attacks can be easily incorporated in our framework. Finally, once an attack is detected, the response unit can provide adequate intrusion response depending upon the security policy. In order to take advantages of Layered framework, each layer must contain both of the above mentioned components.

3.4 Advantages of Layered Framework

We now summarize the advantages of using our layered framework.

1. Using our layered framework improves attack detection accuracy and the system can detect a wide variety of attacks by making use of the domain specific knowledge.

2. The layered framework does not degrade system performance as individual layers are independent and are trained with only a small number of features, thereby, resulting in an efficient system.

   Additionally, using our layered framework opens avenues to perform pipelining resulting in very high speed of operation. Implementing pipelining, particularly in multi core processors, can significantly improve the performance by reducing the multiple I/O operations to a single I/O operation since all the features can be read in a single operation and analyzed by different layers in the layered framework.

3. Our framework is easily customizable and the number of layers can be adjusted depending upon the requirements of the target network.
4. Our framework is not restrictive in using a single method to detect attacks. Different methods can be seamlessly integrated in our framework to build effective intrusion detectors.
5. Layered framework for building effective and efficient network intrusion detection systems fits well in the traditional layered defense approach for providing network and enterprise level security.
6. Our framework has the advantage that the type of attack can be inferred directly from the layer at which it is detected. As a result, specific intrusion response mechanisms can be activated for different attacks.

3.5 Comparison with other Frameworks

Ensuring continuity of services and security of data from unauthorized disclosure and malicious modifications are critical for any organization. However, providing a desired level of security at the enterprise level can be challenging. No single tool can provide enterprise wide security and hence, a number of different security tools are deployed. For this, a layered defense approach is often employed to provide security at the organizational level. This traditional layered defense approach incorporates a variety of security tools such as the network surveillance, perimeter access control, firewalls, network, host and application intrusion detection systems, file integrity checkers, data encryption and others which are deployed at different access points in a layered security framework. The traditional layered architecture is perceived as a framework for ensuring complete organizational security rather than as an approach for building effective and efficient intrusion detection systems. Figure 3.2 represents the traditional layered defense approach. However, as discussed earlier, we present a layered framework for building intrusion detection systems. Our framework fits well in the traditional layered defense approach and can be used to develop effective and efficient network intrusion detection systems. Further, the four components viz., event generators, event Analyzers, event Databases and the response units, presented in the Common Intrusion Detection Framework [20] can be defined for every intrusion detection subsystem in our layered framework.

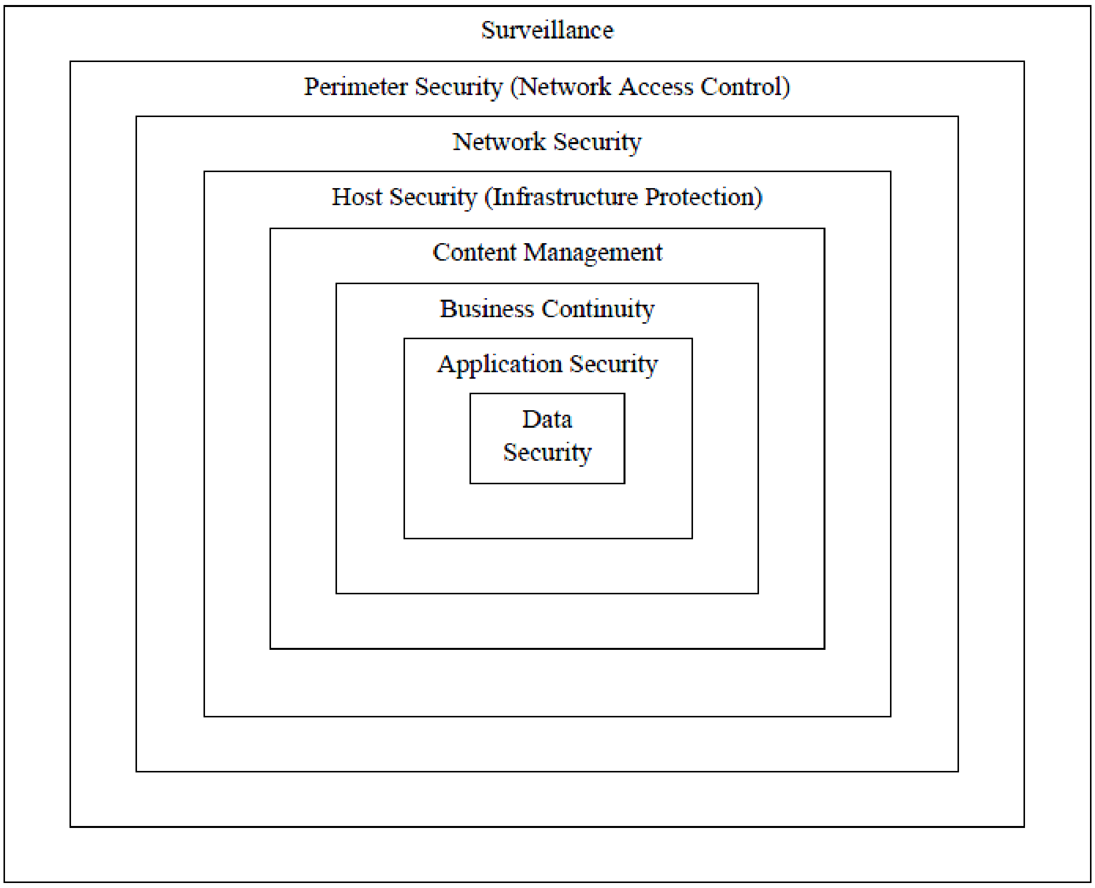


Figure 3.2 Traditional Layered defense approach

In the data mining framework for intrusion detection, [50], the authors describe the use of data mining algorithms to compute activity patterns from system audit data to extract features which are then used to generate rules to detect intrusions. The same approach can be applied for building an intrusion detection system based on our layered framework. Our framework can not only seamlessly integrate the use of data mining technique for intrusion detection, but can also help to improve its performance by selecting only a small number of significant features for building separate intrusion detection sub systems which can be used to effectively detect different classes of attacks at different layers.

A number of other frameworks have been proposed which describe the use of classifier combination [30], [61], [63], [64]. In [30] and [61], the authors apply a combination of anomaly and misuse detectors for better qualification of analyzed events. The authors in [63] describe the combination of 'strong' classifiers using stacking where decision tress, naive Bayes and a number of other classification methods are used as base classifiers. The authors show that the output from these classifiers can be combined to generate a better classifier rather than selecting the individual best classifier. In [64], the authors use a combination of 'weak' classifiers where the individual classification power of weak classifiers is slightly better than that of random guessing. The authors show that a number of such classifiers when combined by using simple majority voting mechanism provide good classification. Our framework is, however, not based upon classifier combination. Combination of classifiers is expensive with regards to the processing time and decision making. In addition, centralized decision making systems often tend to be complex and slow in operation.

The only purpose of classifier combination is to improve accuracy. Rather, our system is based upon serial layering of multiple hybrid detectors which are trained independently and which operate without the influence of any central controller. In our framework, the results from individual classifiers at a layer are not combined at any later stage and, hence, an attack is blocked at the layer where it is detected. There is no communication overhead among the layers and the central decision maker which results in an efficient system. In addition, since the layers are independent they can be trained separately and deployed independently. As already discussed, using a stacked system is expensive when compared to the sequential model. From our experimental results in the following chapters, we will show that an intrusion detection system based on our layered framework performs better and is more efficient when compared with individual systems as well as with systems based on classifier combination.

Chapter 4
Layered Approach Using Conditional Random Fields

In previous chapter we have seen that layered approach is better than centralized approach and provide better results. It can also be used with various techniques. In this chapter we will give the various benefits of using conditional random fields and combine it with the layered approach.

4.1 Conditional Random Fields

Conditional Random Fields are undirected graphical models used for the task of sequence tagging. The difference between the Conditional Random Fields (CRF) and other graphical models such as the Hidden Markov Models (HMM) is that the HMM, being generative, models the joint distribution $p(y, x)$ whereas the CRF are discriminative models directly modeling the conditional distribution $p(y|x)$ which is the distribution of interest. Similar to HMM, the Naive Bayes are also generative and model the joint distribution.

Modeling the joint distribution for the task of classification and sequence labeling has two disadvantages.

1. Since the observations are completely visible so it is not the distribution of interest. The interest is in finding the correct class for the visible observation which is the same as the conditional distribution $p(y|x)$.

2. Inferring the conditional probability $p(y|x)$ from the modeled joint distribution, using the Bayes rule, requires the marginal distribution $p(x)$. To calculate this marginal distribution is difficult as the amount of training data is often limited and the observation x contains highly dependent features which are difficult to model and hence strong independence assumptions are made to simplify the task. This results in reduced accuracy [86].

Thus, the Conditional Random Fields simply try to predict y given the x. This allows them to model arbitrary features in different attributes in x [87]. Conditional Random Fields also avoid the observation bias and the label bias problem which is present in other discriminative models, such as the Maximum Entropy Markov Models. The Maximum Entropy Markov Models use per-state exponential model for the conditional probabilities of the next state given the current state and the observation sequence [88] while the Conditional Random Fields have a single exponential model for the joint probability of the entire sequence of labels given the observation sequence, thus avoiding the label bias problem [89]. Given X and Y , the random variables over data sequence to be labeled and corresponding label sequences, let $G = (V,E)$ be a graph such that $Y = (Y_v)$ is represented by the vertices of the graph G, then, (X, Y) is a Conditional Random Field, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: $p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G, [89], i.e. a CRF is a random field globally conditioned on X. For a simple sequence modeling, as in our case, the joint distribution over the label sequence Y given X has the form:

$$p_\theta(y|x) \propto \exp(\sum_{e \in E,k} \lambda k f k(e, y|e, x) + \sum_{v \in V,k} \mu k g k(v, y|v, x))$$

where x is the data sequence, y is a label sequence, and $y|s$ is the set of components of y associated with the vertices in sub graph S. Also, the features f_k and g_k are assumed to be

given and fixed [89]. The parameter estimation problem is to find the parameters $\theta = (\lambda_1, \lambda_2, ...; \mu_1, \mu_2, ...)$ from the training data $D = (x^i, y^i)_{i=1}^{N}$ with the empirical distribution $\tilde{p}(x, y)$ [89]. The graphical structure of a Conditional Random Fields can be represented as shown in Figure 1

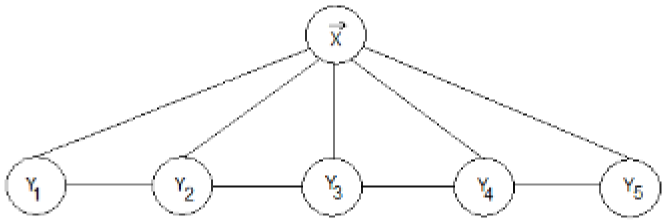


Figure 4.1. Graphical Representation of a Conditional Random Field

where $\vec{X}$ represents a sequence of length five, in this case, and each attribute of $\vec{X}$ is correspondingly labeled as Y_i. The task of Intrusion Detection can now be compared to many problems in Machine Learning, Natural Language Processing and Bio-Informatics. The Conditional Random Fields have been proven to be very successful in such tasks, as they make no unwarranted assumptions about the data, and once trained they also appear to be very efficient and robust. The task of Intrusion Detection, however, has some major requirements. It has to be an online task and there is no knowledge available for the future observations. Further, once deployed, the system has to deal with large amount of data and thus it must be able to perform fast enough to be effective. Conditional Random Fields satisfy all of these requirements and once the model has been trained and deployed, they are very fast in labeling the data as either normal or as attack. The complexity of a Conditional Random Field is quadratic with respect to the number of labels. This is problematic when the number of labels is large, such as in the language tasks, but in our case we have only two labels; normal and attack. We observe that the training of a Conditional Random Field is expensive but once trained their performance is comparable to that of the Decision Trees and Naive Bayes classifiers. Thus our system is very efficient and can be used online. As discussed in Section 2, the current work does not consider the relationships among the attributes in the observations. They either consider only one attribute, such as in the system call modeling, or assume conditional independence among the attributes. However, if we can model these relations suitably, the system is bound to perform better. As we will show from our experimental results, the Conditional Random Fields can be effectively used to model such complex relationships among the attributes of an observation without compromising the accuracy and efficiency of classification. We first perform the sequence labeling using a Conditional Random Field where the system considers every record as a separate sequence of attributes and labels each attribute of this sequence to give the final label for each record. We then perform experiments with the Decision Trees and Naive Bayes classifier and compare the results with the Conditional Random Fields.

4.2. Motivating Examples

It is necessary to search for methods which can be used to hybrid intrusion detection systems. However, given that the present networks are prone to a wide variety of attacks, using a single system would not only degrade performance but will also be less effective in attack detection.

Consider for example, a single network intrusion detection system which is deployed to detect every network attack in a high speed network. A network is prone to different types of attacks such as the Denial of Service (DoS), Probe and others. We note that the DoS and Probe attacks are different and require different features for their effective detection. When same features are used to detect the two attacks the accuracy decreases. It also makes the system bulky which affects its speed of operation. Hence, for effective attack detection, a network intrusion detection system must differentiate between different types of attacks. Thus, using a single system is not a viable option. One possible solution is having a number of sub systems each of which is specific in detecting a single category of attack (such as DoS, Probe and others). This is not only more effective in detecting individual classes of attacks, but it also results in an efficient system. The number of sub systems to be used can be determined by analyzing the potential risks and the availability of resources at individual installations.

For example, a network intrusion detection system may consist of four layers, where the layers correspond to four different attack classes; Denial of Service, Probe, Remote *to* Local and User to Root.

Two main requirements for an intrusion detection system;
1) Accuracy of detection and
2) Efficiency in operation.

The CRFs can be effective in improving the attack detection accuracy by reducing the number of false alarms, while the Layered Approach can be implemented to improve the overall system efficiency. Hence, a natural choice is to integrate them to build a single system that is accurate in detecting attacks and efficient in operation. Given the data, we first select four layers corresponding to the four attack groups (Probe, DoS, R2L, and U2R) and perform feature selection for each layer, which is described next.

4.3 Feature Selection

Ideally, feature selection should be done automatically. However, the methods for automatic feature selection were not found to be effective. In this section, we describe our approach for selecting features for every layer and why some features were chosen over others. In this system, every layer is separately trained to detect a single type of attack category. We observe that the attack groups are different in their impact, and hence, it becomes necessary to treat them differently. Hence, we select features for each layer based upon the type of attacks that the layer is trained to detect.

4.3.1 Probe Layer

The probe attacks are aimed at acquiring information about the target network from a source that is often external to the network. Hence, basic connection level features such as the "duration of connection" and "source bytes" are significant while features like "number of files creations" and "number of files accessed" are not expected to provide information for detecting probes.

4.3.2 DoS Layer

The DoS attacks are meant to force the target to stop the service(s) that is (are) provided by flooding it with illegitimate requests. Hence, for the DoS layer, traffic features such as the "percentage of connections having same destination host and same service" and packet level features such as the "source bytes" and "percentage of packets with errors" are significant. To detect DoS attacks, it may not be important to know whether a user is "logged in or not."

4.3.3 R2L Layer

The R2L attacks are one of the most difficult to detect as they involve the network level and the host level features. We therefore selected both the network level features such as the “duration of connection” and “service requested” and the host level features such as the “number of failed login attempts” among others for detecting R2L attacks.

4.3.4 U2R Layer

The U2R attacks involve the semantic details that are very difficult to capture at an early stage. Such attacks are often content based and target an application. Hence, for U2R attacks, we selected features such as “number of file creations” and “number of shell prompts invoked,” while we ignored features such as “protocol” and “source bytes.” We used domain knowledge together with the practical significance and the feasibility of each feature before selecting it for a particular layer. Thus, from the total 41 features, we selected only 5 features for Probe layer, 9 features for DoS layer, 14 features for R2L layer, and 8 features for U2R layer. Since each layer is independent of every other layer, the feature set for the layers is not disjoint. The selected features for all the four layers are presented in Appendix A. We then use the CRFs for attack detection as discussed in Section 3. However, the difference is that we use only the selected features for each layer rather than using all the 41 features. We now give the algorithm for integrating CRFs with the Layered Approach.

Algorithm
Training

Step 1: Select the number of layers, n, for the complete system.
Step 2: Separately perform features selection for each layer.
Step 3: Train a separate model with CRFs for each layer using the features selected from Step 2.
Step 4: Plug in the trained models sequentially such that only the connections labeled as normal are passed to the next layer.

Testing

Step 5: For each (next) test instance perform Steps 6 through 9.

Step 6: Test the instance and label it either as attack or normal.

Step 7: If the instance is labeled as attack, block it and identify it as an attack represented by the layer name at which it is detected and go to Step 5. Else pass the sequence to the next layer.

Step 8: If the current layer is not the last layer in the system, test the instance and go to Step 7. Else go to Step 9.

Step 9: Test the instance and label it either as normal or as an attack. If the instance is labeled as an attack, block it and identify it as an attack corresponding to the layer name.

Our final goal is to improve both the attack detection accuracy and the efficiency of the system. Hence, we integrate the CRFs and the Layered Approach to build a single system. We perform detailed experiments and show that our integrated system has dual advantage. First, as expected, the efficiency of the system increases significantly. Second, since we select significant features for each layer, the accuracy of the system further increases. This is because all the 41 features are not required for detecting attacks belonging to a particular attack group. Using more features than required can result in fitting irregularities in the data, which has a negative effect on the attack detection accuracy of the system.

4.4 Implementing the System in Real Life

In real scenario, we are not aware of the category of an attack. Rather, we are interested in identifying the attack category once the system detects an event as anomalous. Layered Approach not only improves the attack detection, but it also helps identify the type of attack once detected because every layer is trained to detect only a particular category of attack. Hence, if an attack is detected at the U2R layer, it is very likely that the attack is of “U2R” type. This enables to perform quick recovery and prevent similar attacks. Fig. 4.1 gives the real-time system representation.

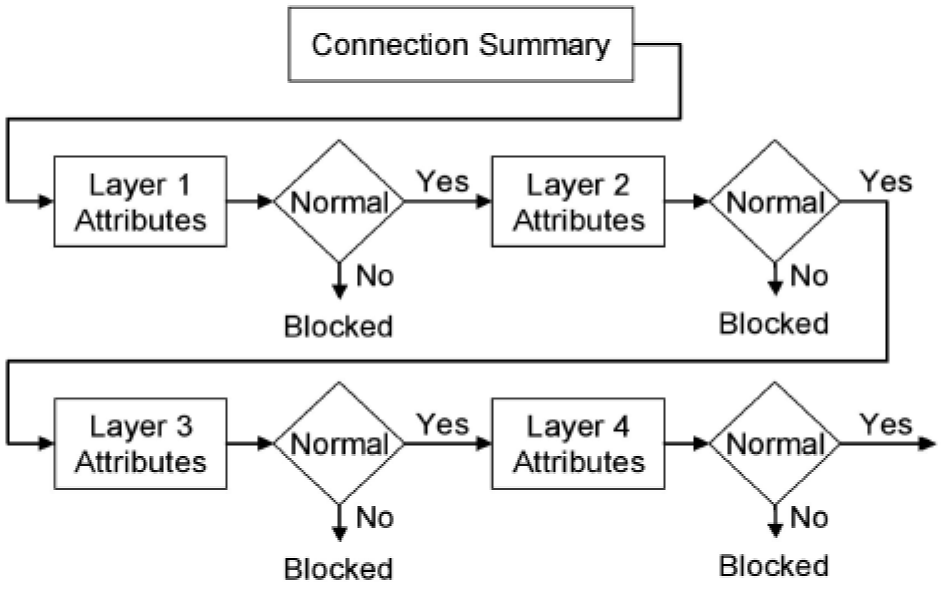


Figure 4.2 Real time systems representation.

We integrate the four models (with feature selection) to develop the final system. In this experiment, we use the same data for training the individual models as used in our previous experiments. However, the data in the test set is relabeled either as normal or as attack and all the data from the test set is passed though the system starting from the first layer. If layer 1 detects any connection as an attack, it is blocked and labeled as "Probe." Only the events labeled as "Normal" are allowed to go to the next layer. The same process is repeated at the next layers where an attack is blocked and labeled as "DoS," "R2L," or "U2R" at layer 2, layer 3, and layer 4, respectively. We perform all the experiments 10 times and report their average.

A number of features are selected at various layers. These features are selected on the basis of the layers and may not be disjoint. So, each layer may include some of the features from the other layer to make it independent to find that types of attack. It also increases the efficiency of each layer. These features include various network and user level features. These features are shown in the table given below:

- 5 features for Probe layer,
- 9 features for DoS layer,
- 14 features for R2L layer, and
- 8 features for U2R layer.

Features to be checked and stored at various Layers.

Feature Number	Feature Name
1	duration
2	protocol_type
3	service
4	Flag
5	scr_bytes

Table 4.1 Features for Probe layer

Feature Number	**Feature Name**
1	Duration
2	protocol_type
4	Flag
5	Count
23	scr_byte
34	dst_host_same_srv_rate
38	dst_host_serror_rate
39	dst_host_srv_serror_rate
40	dst_host_rerror_rate

Table 4.2 Features for DoS layer

Feature Number	Feature Name
1	Duration
2	protocol_type
3	Service
4	Flag
5	scr_bytes
10	Hot
11	num_failed_logins
12	logged_in
13	num_compromised
17	num_file_creation
18	num_shells
19	num_scess_files
21	is_host_login
22	is_guest_login

Table 4.3 Features of R2L Layer

Feature Number	Feature Name
10	Hot
13	num_compromised
14	root_shell
16	num_root
17	num_file_creations
18	num_shells
19	num_access_files
21	is_host_login

Table 4.4 Features for U2R layer.

Chapter 5
Intrusion detection using Artificial Neural Networks and fuzzy clustering

Many researchers have argued that Artificial Neural Networks (ANNs) can improve the performance of intrusion detection systems (IDS) when compared with traditional methods. However for ANN-based IDS, detection precision, especially for low-frequent attacks, and detection stability are still needed to be enhanced. Gang Wang, Jinxing Hao, Jian Ma and Lihua Huang proposed an approach called FC-ANN [68], based on ANN and fuzzy clustering, to solve the problem and help IDS achieve higher detection rate, less false positive rate and stronger stability.

5.1 Introduction

In the early stage, the research focus lies in using rule-based expert systems and statistical approaches [78]. But when encountering larger datasets, the results of rule-based expert systems and statistical approaches become worse. Thus a lot of data mining techniques have been introduced to solve the problem [70]. Among these techniques, Artificial Neural Network (ANN) is one of the widely used techniques and has been successful in solving many complex practical problems. And ANN has been successfully applied into IDS [71].

However, the main drawbacks of ANN-based IDS exist in two aspects:

(1) Lower detection precision, especially for low-frequent attacks, e.g., Remote to Local (R2L), User to Root (U2R),
(2) Weaker detection stability [77].

For the above two aspects, the main reason is that the distribution of different types of attacks is imbalanced. In practice, low-frequent attacks do not mean they are unimportant. Instead, serious consequence will be caused if these attacks succeeded. For example, if the U2R attacks succeeded, the attacker can get the authority of root user and do everything he likes to the targeted computer systems or network device. Furthermore in IDS the low-frequent attacks are often outliers. Thus ANN is unstable as it often converges to the local minimum [69] Although prior research has proposed some approaches, when encountering large datasets, these approaches become not effective [75]. To solve the above two problems, we propose a novel approach for ANN-based IDS, FC-ANN, to enhance the detection precision for low-frequent attacks and detection stability.

5.2 Frame work for FCANN

FC-ANN firstly divides the training data into several subsets using fuzzy clustering technique. Subsequently, it trains the different ANN using different subsets. Then it determines membership grades of these subsets and combines them via a new ANN to get final results. The whole framework of FC-ANN is illustrated in Fig.5.1. As typical machine learning framework, FC-ANN incorporates both the training phase and testing phase. The training phase includes the following three major stages:

Stage I: For an arbitrary data set DS, it is firstly divided into training set TR and testing set TS. Then the different training subsets TR_1; TR_2; . . . ; TR_k are created from TR with fuzzy clustering module.

Stage II: For each training subset TR_i (i= 1; 2; . . . ; k), the ANN model, ANN_i, (i =1; 2; . . . ; k) is training by the specific learning algorithm to formulate k different base ANN models.

Stage III: In order to reduce the error for every ANN_i, we simulate the ANN_i using the whole training set TR and get the results. Then we use the membership grades, which were generated by fuzzy clustering module, to combine the results. Subsequently, we train a new ANN using the combined results. The whole framework is shown in figure 5.1

In the testing phase, we directly input the testing set data into the k different ANN_i and get outputs. Based on these outputs, the final results can then be achieved by the last fuzzy aggregation module.

The three stages of FC-ANN framework raise three important issues:

(1) How to create k different training subsets from the original training dataset TR;
(2) How to create different base model ANN_i with different training subsets;
(3) How to aggregate the different results produced by different base model ANN_i. These issues will be addressed by the following sections, respectively.

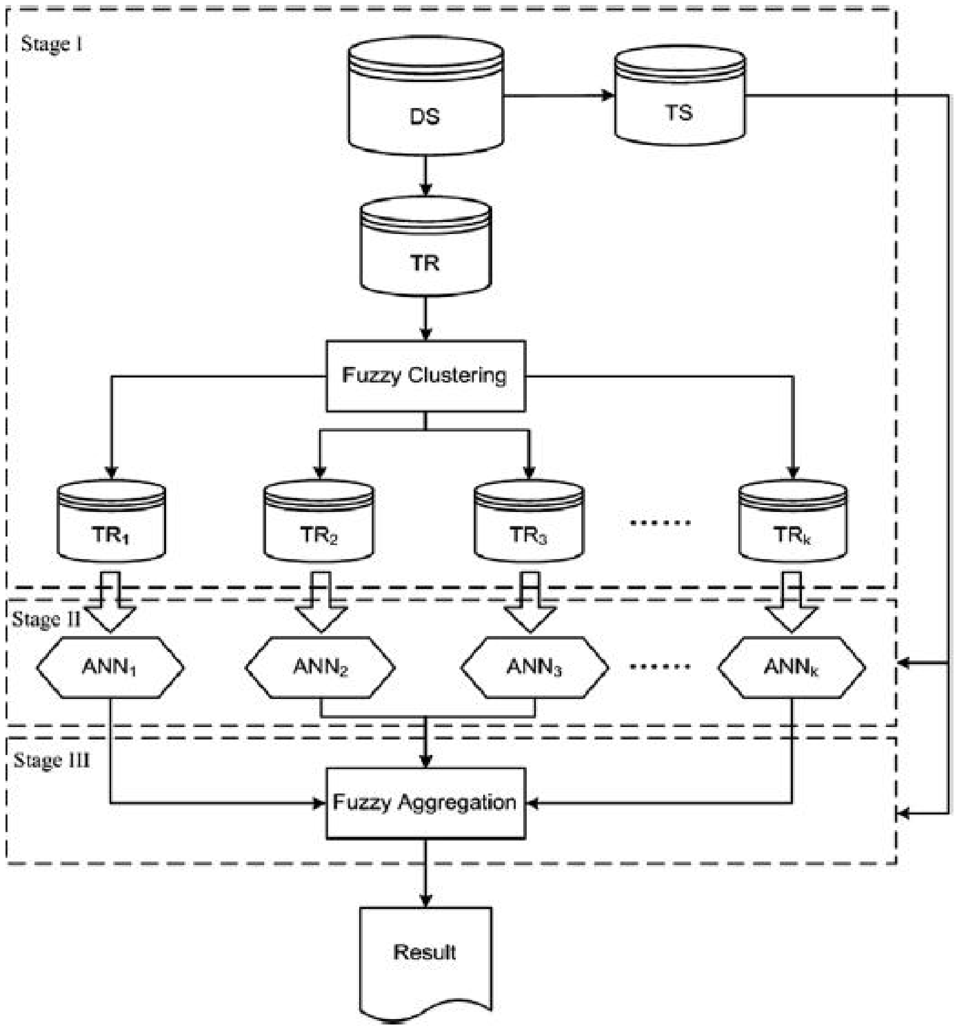


Figure 5.1 Framework of FCANN for IDS

5.2.1 Fuzzy clustering module

The aim of fuzzy cluster module is to partition a given set of data into clusters, and it should have the following properties:

- Homogeneity within the clusters,
- Concerning data in same cluster, and
- Heterogeneity between clusters, where data belonging to different clusters should be as different as possible.

Through fuzzy clustering module, the training set is clustered into several subsets. Due to the fact that the size and complexity of every training subset is reduced, the efficiency and effectiveness of subsequent ANN module can be improved. The clustering techniques can be divided into hard clustering techniques and soft clustering techniques [73]. Beside partition of training set, we also need to aggregate the results for Fuzzy aggregation module. Therefore, we choose one of the popular soft clustering techniques, fuzzy c-means clustering, for fuzzy clustering module [72].

Fuzzy c-means is a data clustering algorithm in which each data point belongs to a clustering to a degree specified by a membership grade [72].In fuzzy clustering module, it is based on the minimization of the following objective function:

1) Homogeneity within the clusters, concerning data in same cluster.
2) Heterogeneity between clusters, where data belonging to different clusters should be as different as possible.

Fuzzy c-means

Fuzzy c-means is a data clustering algorithm in which each data point belongs to a clustering to a degree specified by a membership grade (Chiu, 1994; Yager & Filev, 1994). In fuzzy clustering module, it is based on the minimization of the following objective function

$$J_m^{TR} = \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ij}^{TR^m} \left\| x_i^{TR} - c_j^{TR} \right\|^2 , \quad 1 \quad \leq m < \infty \qquad (5.1)$$

where m is any real number greater than 1, $u_{ij}^{TR}$is the degree of membership of x_i^{TR} i in the cluster j, i is the ith of d-dimensional measured data, c_j^{TR} j is the d-dimensional center of cluster, and $\| * \|$ is any norm expressing the similarity between any measured data and center. Fuzzy partitioning is carried out through an iterative optimization of the object function shown above, with the update of membership $u_{ij}^{TR}$and the cluster centers c_j^{TR} by

$$u_{ij}^{TR} = \frac{1}{\sum_{p=1}^{k} \left(\frac{\left\| x_i^{TR} - c_j^{TR} \right\|}{\left\| x_i^{TR} - c_p^{TR} \right\|} \right)^{\frac{2}{m-1}}} \quad , \quad c_j^{TR} = \frac{\sum_{i=1}^{n} u_{ij}^{TR^m} x_i^{TR}}{\sum_{i=1}^{n} u_{ij}^{TR^m}} \qquad (5.2)$$

This iteration will stop when:

$$\max_{ij} \left\{ \left| u_{ij}^{TR(q+1)} - u_{ij}^{TR(q)} \right| \right\} < \epsilon \qquad (5.3)$$

Where ϵ is a termination criterion between 0 and 1 and q is the iteration steps. Based on the above analysis, the fuzzy cluster module is composed of the following steps:

Step 1: Initialize $U^{TR} = [u_{ij}^{TR}]$ matrix: $u^{TR}(0)$ and q = 1.

Step 2: At q-step: calculate the centers vectors $C^{TR}(q) = [c_j^{TR}]$ with $U^{TR}(q)$

,

$$c_j^{TR} = \frac{\sum_{i=1}^{n} u_{ij}^{TR^m} x_i^{TR}}{\sum_{i=1}^{n} u_{ij}^{TR^m}} \tag{5.4}$$

Step 3: Update U(q + 1)

$$u_{ij}^{TR} = \frac{1}{\sum_{p=1}^{k}\left(\frac{\|x_i^{TR} - c_j^{TR}\|}{\|x_i^{TR} - c_p^{TR}\|}\right)^{\frac{2}{m-1}}} \tag{5.5}$$

Step 3: If $\|U^{TR}(q+1) - U^{TR}(q)\| < \epsilon$ then Step 4; otherwise return to Step 2.

Step 4: Based on argmax(u_{ij}^{TR}), every individual sample of TR can be allocated into subsets TR_k

After completing the above four steps, the training set TR can be divided into k subsets TR_k. Subsequently, ANN_i is needed to train using these subsets TR_k. Next section, we will discuss how to create different base model ANN_i with different training subset TR_k.

5.2.2 ANN module

ANN module aims to learn the pattern of every subset. ANN is a biologically inspired form of distributed computation [69]. It is composed of simple processing units, and connections between them. In this study, we will employ classic feed-forward neural networks trained with the back-propagation algorithm to predict intrusion. A feed-forward neural network has an input layer, an output layer, with one or more hidden layers in between the input and output layer. The ANN functions as follows: each node i in the input layer has a signal x_i as network's input, multiplied by a weight value between the input layer and the hidden layer. Each node j in the hidden layer receives the signal In(j) according to:

$In(\mathrm{j}) = \theta_j + \sum_{i=1}^{n} x_i w_{ij}$ (5.6)

Then it is passed through the bipolar sigmoid activation function:

$\mathrm{F(x)} = \frac{2}{(1+\exp(-x))} - 1$ (5.7)

The output of the activation function f(In(j)) is then broadcast all of the neurons to the output layer:

$y_k = \theta_k + \sum_{j=1}^{m} w_{jk} f(\ln(j))$ (5.8)

Where h_j and h_k are the biases values in the hidden layer and the output layer respectively. The output value will be compared with the target; in this study, we used the mean absolute error as error function:

$E_m = \frac{1}{2n} \sum_k \sqrt{(T_k - Y_k)^2}$ (5.9)

when n is the number of training patterns, Y_k and T_k are the output value and the target value, respectively. The gradient descent method searches for the global optimum of the network weights, and partial derivatives $\partial E / \partial w$ are computed for each weight in the network. And the weight will adjust according to the expression:

$w(t+1) = w(t) - \eta\, \partial E(t) / \partial w(t)$ (5.10)

Where t is the number of epochs, g is the learning rate. To accelerate the convergence of the error in the learning procedure, the momentum with the momentum gain, a, is includes into Eq. (5.10) (Anderson, 1995):

$w(t+1) = w(t) - \eta\, \partial E(t) / \partial w(t) + \alpha\, \Delta w(t)$ (5.11)

Where the value for a is within 0 and 1. Based on the feed-forward neural networks trained with the back-propagation algorithm, every ANN$_i$ can complete training using different subsets TR$_k$. However, next question is how to aggregate the different results produced by different base model ANN$_{i.}$

5.2.3 Fuzzy aggregation module

The aim of fuzzy aggregation module is to aggregate different ANN's result and reduce the detection errors as every ANN$_i$ in ANN module only learns from the subset TR$_i$. Because the errors are nonlinear, in order to achieve the objective, we use another new ANN to learn the errors as follows:

Step 1: Let the whole training set TR as data to input the every trained ANN$_i$ and get the outputs:

Y_j^{TR} =$[Y_1^{TR}, Y_2^{TR}, \ldots\ldots, Y_{jk}^{TR}]$, j=1,2,…..,n (5.12)

Where n is the number of training set: TR, Y_{jk}^{TR} is the output of ANNk.

Step 2: Form the input for new ANN:

Y_{input}=$[Y_1^{TR}.U_1^{TR}, Y_2^{TR}.U_2^{TR}, \ldots\ldots, Y_n^{TR}.U_n^{TR}]$ (5.13)

Where U_n^{TR} is the membership grade of TR$_n$ belonging to C_{TR}.

Step 3: Train the new ANN. We can use Y$_{input}$ as input and use the whole training set TR's class label as output to train the new ANN. Through above three steps, the new ANN can learn the errors which caused by the individual ANN$_i$ in ANN module. During the stage of testing, work procedure of ANN module and fuzzy aggregation module is similar to the above. Firstly we calculate the membership grade, based on the cluster centers C$_{TR}$. For a new input x_i^{TS} coming, firstly based on C^{TR}, the membership U^{TS} can be calculated by:

$$u_{ij}^{TR} = \frac{1}{\sum_{p=1}^{k}\left(\frac{\|x_i^{TR}-c_j^{TR}\|}{\|x_i^{TR}-c_p^{TR}\|}\right)^{\frac{2}{m-1}}} \qquad (5.14)$$

Then, respectively, using ANN module and fuzzy aggregation module, the output, Y_{output}^{TS}, can be gotten.

5.3 Data preparation

In the experiments, KDD CUP 1999 dataset is used (KDD dataset, 1999)[79]. The KDD CUP 1999 dataset is a version of the original 1998 DARPA intrusion detection evaluation program, which is prepared and managed by the MIT Lincoln Laboratory. The dataset contains about five million connection records as training data and about two million connection records as test data. And the dataset includes a set of 41 features derived from each connection and a label which specifies the status of connection records as either normal or specific attack type. These features have all forms of continuous, discrete, and symbolic variables, with significantly varying ranges falling in four categories: (1) the first category consists of the intrinsic features of a connection, which include the basic features of individual TCP connections. The duration of the connection, the type of the protocol (TCP, UDP, etc.), and network service (http, telnet, etc.) are some of the features. (2) The content features within a connection suggested by domain knowledge are used to assess the payload of the original TCP packets, such as the number of failed login attempts. (3) the same host features examine established connections in the past two seconds that have the same destination host as the current connection, and calculate the statistics related to the protocol behavior, service, etc. (4) the similar same service features inspect the connections in the past two seconds that have the same service as the current connection. Likewise, attacks fall into four categories: (1) Denial of Service (DoS): making some computing or memory resources too busy to accept legitimate users access these resources. (2) Probe (PRB): host and port scans to gather information or find known vulnerabilities. (3) Remote to Local (R2L): unauthorized access from a remote machine in order to exploit machine's vulnerabilities. (4) User to Root (U2R): unauthorized access to local super user (root) privileges using system's susceptibility. Random selection has been used in many applications to reduce the size of the dataset. In this study, we randomly select 18,285 records, similar to prior research [77]. The PRB, R2L, and U2R attack classes were totally selected because of their low portion.

5.4. Evaluation criteria

The following measurements are often proposed to evaluate the detection precision of IDS [81]: true positives, true negatives, false positives, and false negatives.

A **true positive** indicates that the intrusion detection system detects precisely a particular attack having occurred.

A **true negative** indicates that the intrusion detection system has not made a mistake in detecting a normal condition.

A **false positive** indicates that a particular attack has been detected by the intrusion detection system but that such an attack did not actually occur.

A false positive is often produced due to loose recognition conditions, a limitation on detection methods in the intrusion detection system or phenomena caused by particular environmental factors. It represents the accuracy of the detection system. If it is consistently high, this will lead to administrators intentionally ignoring system warnings, and thus allow the system to remain in a dangerous status.

A **false negative** indicates that the intrusion detection system is unable to detect the intrusion after a particular attack has occurred. This is probably caused by a shortage of information about an intrusion type or by the recognition information about such an intrusion event having been excluded from the intrusion detection system. This reveals the completeness of the detection system.

However as the number of instance for the U2R, PRB, and R2L attacks in the training set and test set is every low, these quantities is not sufficient as a standard performance measure (Dokas et al.,2002). Hence, if we use these quantities as a measure for testing the performance of the systems, it could be biased. For these reasons, we give the precision, recall, and F-value which are not dependent on the size of the training and the testing samples. They are defined as follows:

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (5.15)$$

$$\text{Recall} = \frac{TP}{TP+FN} \qquad (5.16)$$

$$\text{F-value} = \frac{(1+\beta^2)*Recall*Precision}{\beta^2*(Recall+Precision)} \qquad (5.17)$$

Where TP, FP, and FN are the number of true positives, false positives and false negatives, respectively, and b corresponds to the relative importance of precision versus recall and is usually set to 1.

5.5 INTRUSION DETECTOR LEARNING

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections.

The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest uses a version of this dataset.

Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

Attacks fall into four main categories:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local super user (root) privileges, e.g., various ``buffer overflow'' attacks;
- Probing: surveillance and other probing, e.g., port scanning.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the "signature" of known attacks can be sufficient to catch novel variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.

5.5.1 DERIVED FEATURES

Stolfo et al. defined higher-level features that help in distinguishing normal connections from attacks. There are several categories of derived features.

The ``same host'' 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.

The similar ``same service" features examine only the connections in the past two seconds that have the same service as the current connection.

"Same host" and "same service" features are together called time-based traffic features of the connection records.

Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, 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. This yields a set of so-called host-based traffic features.

Unlike most of the DOS and probing attacks, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks. This is because the DOS and probing attacks involve many connections to some host(s) in a very short period of time, but the R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection.

Useful algorithms for mining the unstructured data portions of packets automatically are an open research question. Stolfo et al. used domain knowledge to add features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These features are called ``content" features.

A complete listing of the set of features defined for the connection records is given in the three tables below. The data schema of the contest dataset is available in machine readable form.

feature name	*description*	*Type*
duration	length (number of seconds) of the connection	Continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	Discrete
service	network service on the destination, e.g., http, telnet, etc.	Discrete
src_bytes	number of data bytes from source to destination	Continuous
dst_bytes	number of data bytes from destination to source	Continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
wrong_fragment	number of ``wrong'' fragments	Continuous
urgent	number of urgent packets	Continuous

Table5.1: Basic features of individual TCP connections.

feature name	*description*	*Type*
hot	number of ``hot" indicators	Continuous
num_failed_logins	number of failed login attempts	Continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	number of ``compromised" conditions	Continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if ``su root" command attempted; 0 otherwise	Discrete
num_root	number of ``root" accesses	Continuous
num_file_creations	number of file creation operations	Continuous
num_shells	number of shell prompts	Continuous
num_access_files	number of operations on access control files	Continuous
num_outbound_cmds	number of outbound commands in an ftp session	Continuous
is_hot_login	1 if the login belongs to the ``hot" list; 0 otherwise	Discrete
is_guest_login	1 if the login is a ``guest" login; 0 otherwise	Discrete

Table5.2: Content features within a connection suggested by domain knowledge.

feature name	*description*	*type*
count	number of connections to the same host as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN" errors	continuous
rerror_rate	% of connections that have ``REJ" errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN" errors	continuous
srv_rerror_rate	% of connections that have ``REJ" errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 3: Traffic features computed using a two-second time window.

Chapter 6
EXPERIMENTAL RESULTS

Intrusion detection is one of the critical tasks for networks and applications. We have studied two of the latest approaches for intrusion detection. First approach uses the conditional random fields for the probability calculation while the second approach uses Fuzzy cluttering with artificial neural networks. In this chapter, we will describe the implementation of the both the approaches. We will compare the results of both the approaches.

6.1 Experimental Setup

To evaluate the performance of both the approaches, a series of experiments on KDD CUP 1999 dataset were conducted. In these experiments, we implemented and evaluated the methods in Matlab 2010a on a Windows XP PC with Duo-Core 1.60 GHz CPU and 2.5 GB RAM.

6.2 Implementation of Layered Approach to intrusion Detection Using Conditional Random Fields

The complete procedure has been divided into three sub activities:

1. Preparation of the Data set
2. Training of the IDS
3. Testing of the IDS

6.2.1 Preparation of the Data set

We have used the 10% of the KDD CUP 1999 dataset. Feature selection is one of the most critical task for each layer. Different features are required to train the different layers. As it is a hybrid approach we have used both the normal and attack data for each layer separately as shown in figure 6.1

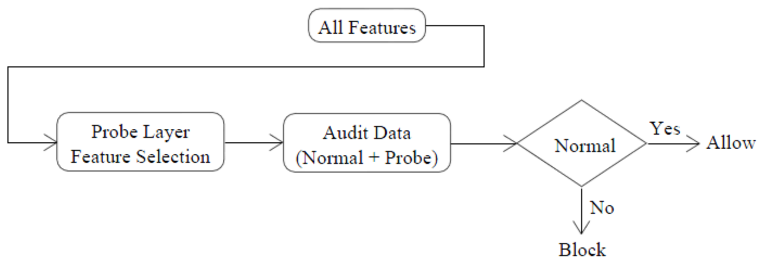


Figure 6.1 Representation of Probe Layer with feature selection and Audit data

We have tagged the data into two forms

1. Attack
2. Normal

The complete data is present under the attack folder.

6.2.2 Training of IDS

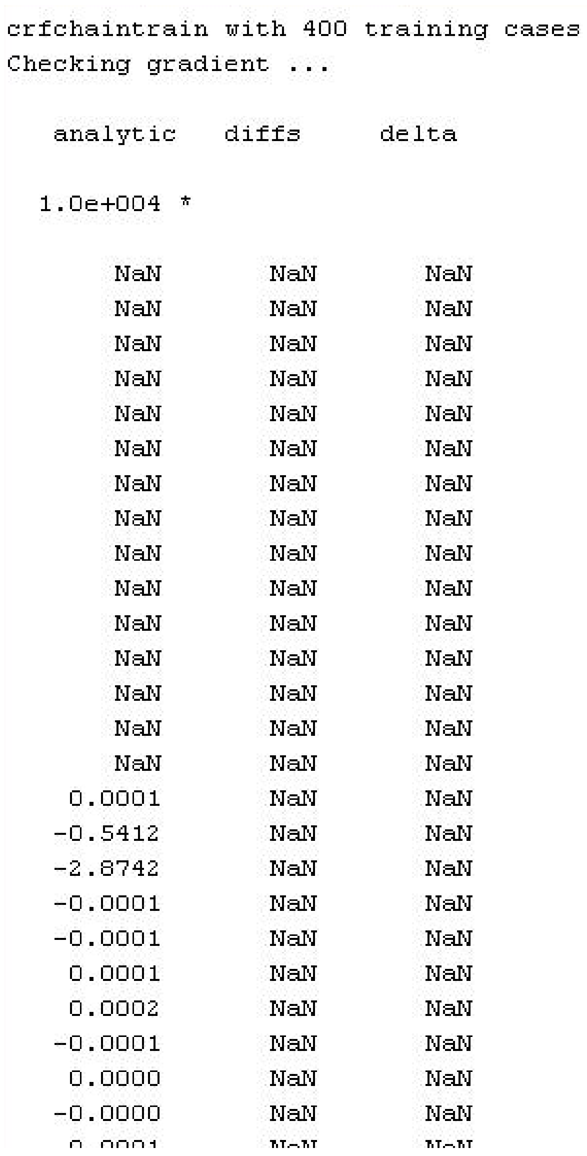
To train with CRF and data set we have used an open source CRF++. Individual layer is trained separately as shown in the figure 6.2. We have used four files to train the system

1) NormalAndProbTrain.m

2)NormalAndDosTrain.m

3)NormalAndRLATrain.m

4) NormalAndURATrain.m

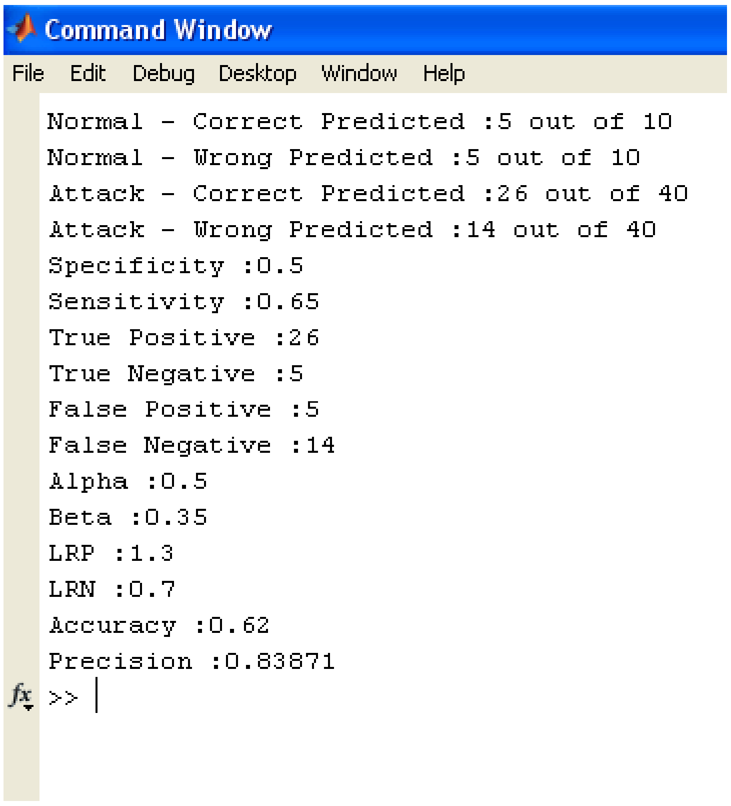


```
crfchaintrain with 400 training cases
Checking gradient ...

   analytic   diffs     delta

  1.0e+004 *

       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
       NaN        NaN        NaN
    0.0001        NaN        NaN
   -0.5412        NaN        NaN
   -2.8742        NaN        NaN
   -0.0001        NaN        NaN
   -0.0001        NaN        NaN
    0.0001        NaN        NaN
    0.0002        NaN        NaN
   -0.0001        NaN        NaN
    0.0000        NaN        NaN
   -0.0000        NaN        NaN
```

Figure 6.2 Training of probe layer

6.2.3 Testing of IDS

We have created a file of 50 records which include 10 normal and 40 attack data. We run CRFTesting.m to check the performance of the system as shown in figure 6.3



```
Normal - Correct Predicted :5 out of 10
Normal - Wrong Predicted :5 out of 10
Attack - Correct Predicted :26 out of 40
Attack - Wrong Predicted :14 out of 40
Specificity :0.5
Sensitivity :0.65
True Positive :26
True Negative :5
False Positive :5
False Negative :14
Alpha :0.5
Beta :0.35
LRP :1.3
LRN :0.7
Accuracy :0.62
Precision :0.83871
>>
```

Figure 6.3 Results of Layered approach using Conditional random fields.

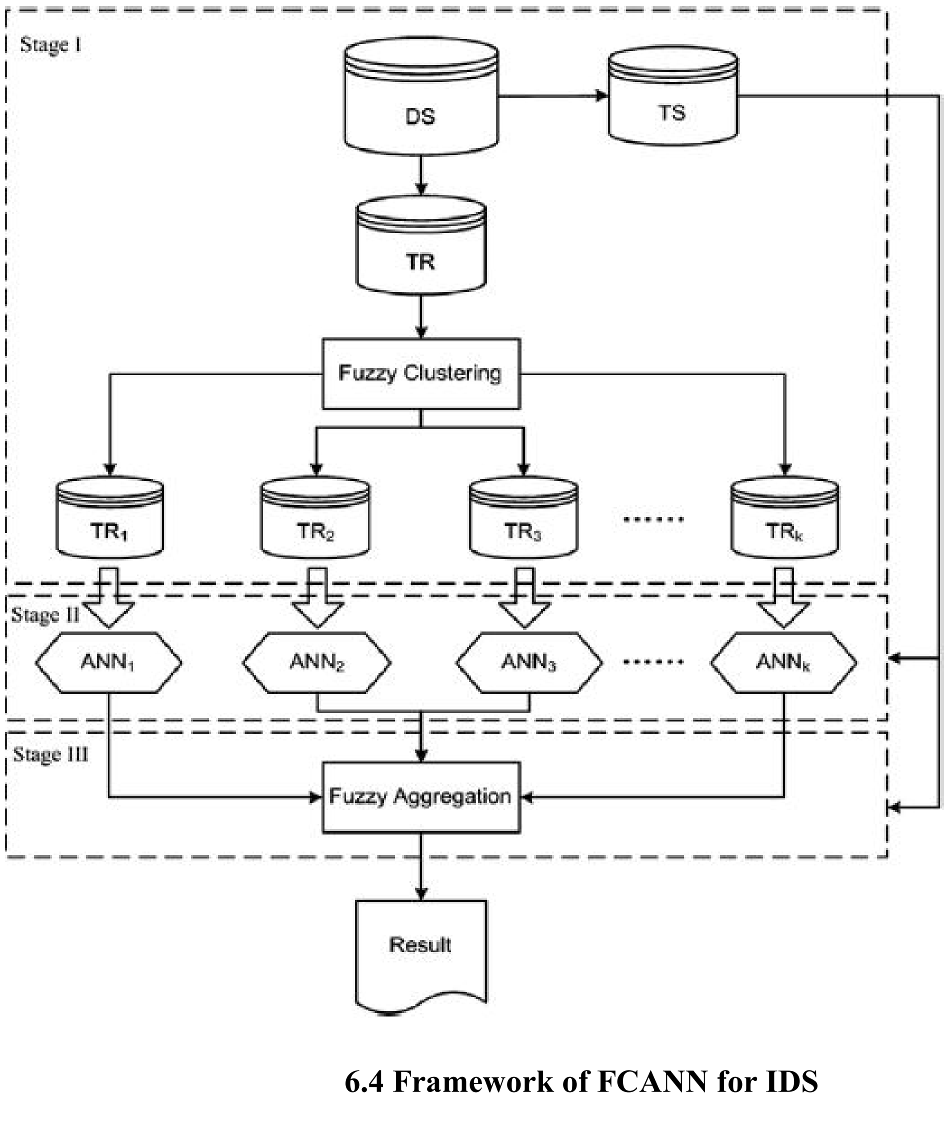
6.3 Implementation of FCANN

FCANN is an artificial neural network technique which uses Fuzzy cluttering to make the training subsets for the individual neural network elements. Framework for the FCANN is shown in the figure 6.4.

6.3.1 Fuzzy Cluttering Module

From the complete data set we first selected the dataset used for the testing and training. The training data is provided as the input to the Fuzzy cluttering module which prepares the dataset for individual neural network. This training set consists of both the kind of data normal as well as attack. We have created "findCluster.m" to create these data set. These datasets are saved in the following matrices.

- TrainDosconData.mat
- TrainProbconData.mat
- TrainRLAconData.mat
- TrainURAconData.mat
- TrainNorconData.mat



6.4 Framework of FCANN for IDS

6.3.2 ANN module

We have used Levenberg-Marquardt (trainlm) for training of Artificial Neural Network which is a classic feed-forward neural networks trained with the back-propagation algorithm to predict intrusion. A feed-forward neural network has an input layer, an output layer, with one or more hidden layers in between the input and output layer. The ANN functions as follows: each node i in the input layer has a signal x_i as network's input, multiplied by a weight value between the input layer and the hidden layer. Each node j in the hidden layer receives the signal In(j) according to:

$$\mathrm{In}(j) = \theta_j + \sum_{i=1}^{n} x_i w_{ij}$$

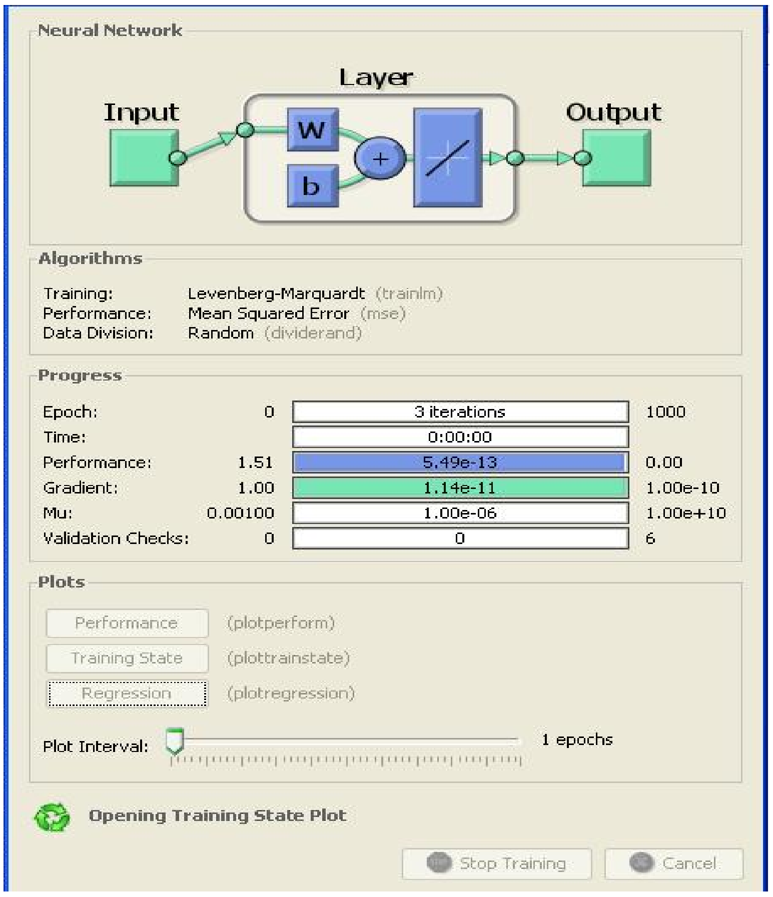
It gives a Mean squared Error performance with random data division. We have used 2 validation checks.

6.3.3 Fuzzy aggregation module

Fuzzy aggregation module aggregates different ANN's result and reduce the detection errors as every ANN_i in ANN module only learns from the subset TR_i. NeuralNetworkTest.m perform the aggregation for net1, net2, net3, net4.

6.3.4 Testing of IDS

Testing.m is run for testing the IDS with the dataset TD which consist of 10 normal records and 40 attack records. The output is shown in the figure 6.6



Algorithms

Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Data Division: Random (dividerand)

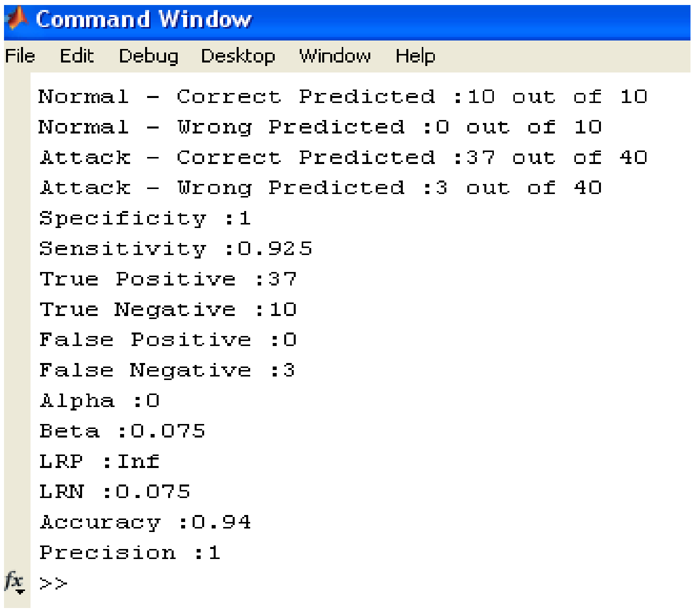
Progress

Epoch:	0	3 iterations	1000
Time:		0:00:00	
Performance:	1.51	5.49e-13	0.00
Gradient:	1.00	1.14e-11	1.00e-10
Mu:	0.00100	1.00e-06	1.00e+10
Validation Checks:	0	0	6

Plot Interval: 1 epochs

Opening Training State Plot

Figure 6.5 Training of IDS



```
Normal - Correct Predicted :10 out of 10
Normal - Wrong Predicted :0 out of 10
Attack - Correct Predicted :37 out of 40
Attack - Wrong Predicted :3 out of 40
Specificity :1
Sensitivity :0.925
True Positive :37
True Negative :10
False Positive :0
False Negative :3
Alpha :0
Beta :0.075
LRP :Inf
LRN :0.075
Accuracy :0.94
Precision :1
>>
```

Figure 6.6 Output of FCANN

6.4 Conclusion and Future Scope

We have observed that FCANN provides better precision and accuracy as compared to conditional random field. Intrusion Detection is one of the major tasks in networks and application. It gives no margin of error. The effective cost of a successful intrusion overshadows the cost of developing intrusion detection system. IDS offers the potential advantages of reducing the manpower needed in monitoring, increasing detection efficiency, providing data that would otherwise not be available, helping the information security community learn about new vulnerabilities and providing legal evidence.

Another possible direction for future research is to employ our approach, layered framework, for building highly efficient systems since they give opportunity to implement pipelining of layers in multi core processors.

Moreover, other data mining techniques, such as support vector machine, evolutionary computing, outlier detection, may be introduced into IDS. Comparisons of various data mining techniques will provide clues for constructing more effective hybrid ANN for intrusions detection.

There is ample scope and need to build systems which aim at preventing attacks rather than simply detecting them. Integrating intrusion detection systems with the security policy in individual networks would help to minimize the false alarms and qualify the alarms raised by the intrusion detection systems.

BIBLIOGRAPHY

[1] Stefan Axelsson. Research in Intrusion-Detection Systems: A Survey. Technical Report 98-17, Department of Computer Engineering, Chalmers University of Technology, 1998.

[2] SANS Institute - Intrusion Detection FAQ http://www.sans.org/resources/idfaq/.

[3] CERT/CC Statistics. http://www.cert.org/stats/

[4] Thomas A. Longstaff, James T. Ellis, Shawn V. Hernan, Howard F. Lipson, Robert D.Mcmillan, Linda Hutz Pesante, and Derek Simmel. Security of the Internet. Technical Report. The Froehlich/Kent Encyclopedia of Telecommunications Vol (15), CERT Coordination Center, 1997. http://www.cert.org/encyc_article/tocencyc.html

[5] Kapil Kumar Gupta, Baikunth Nath, Kotagiri Ramamohanarao, and Ashraf Kazi. Attacking Confidentiality: An Agent Based Approach. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics*, pages 285–296. Lecture Notes in Computer Science, Springer Verlag, Vol (3975), 2006.

[6] James P. Anderson. Computer Security Threat Monitoring and Surveillance, 1980. http://csrc.nist.gov/publications/history/ande80.pdf.

[7] Dorothy E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987. IEEE.

[8] Paul Innella. The Evolution of Intrusion Detection Systems, 2001. http://www.securityfocus.com/infocus/1514

[9] Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt. Network Intrusion Detection. *IEEE Network*, 8(3):26–41, 1994. IEEE.

[10] Herv´e Debar, Marc Dacier, and AndreasWespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(9):805–822, 1999. Elsevier.

[11] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A Sense of Self for Unix Processes. In *Proceeding of the IEEE Symposium on Research in Security and Privacy*, pages 120–128. IEEE, 1996.

[12] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting Intrusions Using System Calls: Alternative Data Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 133–145. IEEE, 1999.

[13] Kapil Kumar Gupta, Baikunth Nath, and Kotagiri Ramamohanarao. Layered Approach using Conditional Random Fields for Intrusion Detection. *IEEE Transactions on Dependable and Secure Computing*, In Press.

[14] Kapil Kumar Gupta, Baikunth Nath, and Kotagiri Ramamohanarao. Robust Application Intrusion Detection using User Session Modeling. *ACM Transactions on Information and Systems Security*, Under Review.

[15] Christopher Kruegel, Fredrik Valeur, and Giovanni Vigna. *Intrusion Detection and Correlation: Challenges and Solutions*. Springer, 2005.

[16] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.

[17] Joseph S. Sherif and Tommy G. Dearmond. Intrusion Detection: Systems and Models. In *Proceedings of the Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE*, pages 115–133. IEEE, 2002.

[18] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[19] Kymie Tan. *Defining the Operational Limits of Sequence-Based Anomaly Detectors*. PhD thesis, The University of Melbourne, 2002.

[20] Stuart Staniford-Chen, Brian Tung, Phil Porras, Cliff Kahn, Dan Schnackenberg, Rich Feiertag, and Maureen Stillman. The Common Intrusion Detection Framework - Data Formats, March 1998. http://tools.ietf.org/html/draft-staniford-cidf-data-formats-00.

[21] Giovanni Vigna and Richard A. Kemmerer. NetSTAT: A Network-based Intrusion Detection Approach. In *Proceedings of the 14th Annual Computer Security Applications Conference*, pages 25–34. IEEE, 1998.

[22] Carol Taylor and Jim Alves-Foss. An Empirical Analysis of NATE: Network Analysis of Anomalous Traffic Events. In *Proceedings of the 2002 Workshop on New Security Paradigms*, pages 18–26. ACM, 2002.

[23] Snort, a Network based Intrusion Detection System http://www.snort.org/

[24] Animesh Patcha and Jung-Min Park. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Computer Networks*, 51(12):3448– 3470, 2007.

[25] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, and Pang-Ning Tan. Data Mining for Network Intrusion Detection. In *Proceedings of the NSF Workshop on Next Generation Data Mining*, pages 21–30, 2002.

[26] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. A Data Mining Framework for Building Intrusion Detection Model. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 120–132. IEEE, 1999.

[27] Dalila Boughaci, Habiba Drias, Ahmed Bendib, Youcef Bouznit, and Belaid Benhamou. Distributed Intrusion Detection Framework Based on Mobile Agents. In *Proceedings of the International Conference on Dependability of Computer Systems*, pages 248–255. IEEE, 2006.

[28] Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene H.Spafford, and Diego Zamboni. Architecture for Intrusion Detection Using Autonomous Agents. In *Proceeding of the 14th Annual Computer Security Applications Conference*, pages 13–24. IEEE, 1998.

[29] Yu-Sung Wu, Bingrui Foo, Yongguo Mei, and Saurabh Bagchi. Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 234–244. IEEE, 2003.

[30] Elvis Tombini, Herv´e Debar, Ludovic Me, and Mireille Ducasse. A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic. In *Proceedings of the 20th Annual Computer Security Applications Conference*, pages 428–437. IEEE, 2004.

[31] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion Detection with Unlabeled Data using Clustering. In *Proceedings of the ACM Workshop on Data Mining Applied to Security (DMSA)*. ACM, 2001.

[32] H. Shah, J. Undercoffer, and A. Joshi. Fuzzy Clustering for Intrusion Detection. In *Proceedings*
of the 12th IEEE International Conference on Fuzzy Systems, pages 1274–1278. IEEE, 2003.

[33] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 207–216. ACM, 1993.

[34] H.Mannila, H.Toivonen, and A.I.Verkamo. Discovering Frequent Episodes in Sequences. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, pages 210–215. AAAI, 1995.

[35] Nahla Ben Amor, Salem Benferhat, and Zied Elouedi. Naive Bayes vs Decision Trees in Intrusion Detection Systems. In *Proceedings of the ACM Symposium on Applied Computing*, pages 420–424. ACM, 2004.

[36] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, 1997. Springer. [37] Darren Mutz, Fredrik Valeur, Giovanni Vigna, and Christopher Kruegel. Anomalous System Call Detection. *ACM Transactions on Information and System Security*, 9(1):61–93,
2006. ACM.

[38] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. Bayesian Event Classification for Intrusion Detection. In *Proceedings of 19th Annual Computer Security Applications Conference*, pages 14–23. IEEE, 2003.

[39] Gray Stein, Bing Chen, Annie S. Wu, and Kien A. Hua. Decision Tree Classifier for Network Intrusion Detection with GA-Based Feature Selection. In *Proceedings of the 43rdAnnual South East Regional Conference - Volume 2*, pages 136–141. ACM, 2005.

[40] Srinivas Mukkamala, Guadalupe Janoski, and Andrew H. Sung. Intrusion Detection Using Neural Networks and Support Vector Machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1702–1707. IEEE, 2002.

[41] Andrew H. Sung and Srinivas Mukkamala. Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In *Proceedings of Symposium on Applications and the Internet*, pages 209–216. IEEE, 2003.

[42] Dong Seong Kim and Jong Sou Park. Network-Based Intrusion Detection with Support Vector Machines. In *Proceedings of the Information Networking, Networking Technologies for Enhanced Internet Services International Conference, ICOIN*, pages 747–756. Lecture Notes in Computer Science, Springer Verlag, 2003.

[43] S. Jha, K. Tan, and R.A. Maxion. Markov chains, Classifiers, and Intrusion Detection. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 206–219. IEEE, 2001.

[44] Nong Ye, Yebin Zhang, and Connie M. Borror. Robustness of the Markov-Chain Model for Cyber-Attack Detection. *IEEE Transactions on Reliability*, 53(1):116–123, 2004.

[45] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[46] Svetlana Radosavac. Detection and Classification of Network Intrusions using Hidden Markov Models. Master's thesis, University of Maryland, 2003.

[47] Wei Wang, Xiao-Hong Guan, and Xiang-Liang Zhang. Modeling Program behaviors by Hidden Markov Models for Intrusion Detection. In *Proceedings of International Conference on Machine Learning and Cybernetics*, pages 2830–2835. IEEE, 2004.

[48] Ye Du, Huiqiang Wang, and Yonggang Pang. A Hidden Markov Models-Based Anomaly Intrusion Detection Method. In Proceeedings of the Fifth World Congress on Intelligent Control and Automation (WCICA), pages 4348–4351. IEEE, 2004.

[49] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, 2001.

[50] Wenke Lee and Salvatore J. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):227–261, 2000. ACM.

[51] CERT/CC Statistics. http://www.cert.org/stats/.

[52] L. T. Heberlein, G.V. Dias, K. N. Levitt, B. Mukherjee, J.Wood, and D.Wolber. A Network Security Monitor. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 296–304. IEEE, 1990.

[53] Rebecca Bace and Peter Mell. *Intrusion Detection Systems*. Gaithersburg, MD : Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, 2001.

[54] H. S. Javitz and A. Valdes. The SRI IDES Statistical Anomaly Detector. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 316–326. IEEE, 1991.

[55] Adwait Ratnaparkhi. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, 1996.

[56] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

[57] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, 1996.

[58] Andrew McCallum, Dyane Freitag, and Fernando Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598. Morgan Kaufmann, 2000.

[59] Dan Klein and Christopher D. Manning. Conditional Structure versus Conditional Estimation in NLP Models. In *Proceedings of the ACL-02 Conference on Empirical methods in Natural Language Processing Vol (10)*, pages 9–16. Association for Computational Linguistics, 2002.

[60] Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields for Relational Learning. In *Introduction to Statistical Relational Learning*. MIT, 2006.

[61] L. Ertoz, A. Lazarevic, E. Eilertson, Pang-Ning Tan, Paul Dokas, V. Kumar, and Jaideep Srivastava. Protecting Against Cyber Threats in Networked Information Systems. In *Proceedings of SPIE; Battlespace Digitization and Network Centric Systems III*, pages 51–56, 2003.

[62] Shon Harris. *CISSP All-in-One Exam Guide*. McGraw-Hill Osborne Media, 2007.

[63] Saso Dzeroski and Bernard Zenko. Is Combining Classifiers Better than Selecting the Best One. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 123–129. Morgan Kaufmann, 2002.

[64] Chuanyi Ji and Sheng Ma. Combinations of Weak Classifiers. *IEEE Transactions on Neural Networks*, 8(1):32–42, 1997.

[65] Andrew Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[66]Yu Gu, Andrew McCallum, and Don Towsley. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. In *Proceedings of the Internet Measurement Conference*, pages 345–350. USENIX Association, 2005.

[67]Layered Approach Using Conditional Random Fields for Intrusion Detection by Kapil Kumar Gupta, Baikunth Nath, Senior Member and Ramamohanarao Kotagiri, IEEE, 2010

[68]A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering by Gang Wang , Jinxing Hao , Jian Ma, Lihua Huang, ELSEVIER, 2010.

[69]Haykin, S. (1999). Neural networks: A comprehensive foundation. Prentice Hall. Manikopoulos, C., & Papavassiliou, S. (2002). Network intrusion and fault detection: A statistical anomaly approach. IEEE Communications Magazine, 40(10), 76–82.

[70]Dokas, P., Ertoz, L., Lazarevic, A., Srivastava, J., & Tan, P. N. (2002). Data mining for network intrusion detection. Proceeding of NGDM, 21–30.

[71]Ryan, J., Lin, M., & Miikkulainen, R. (1998). Intrusion detection with neural networks. Advances in neural information processing systems (Vol. 10). Cambridge, MA: Springer

[72]Yager, R. R., & Filev, D. P. (1994). Approximate clustering via the mountain method. IEEE Transactions on Systems, Man and Cybernetics, 24(8), 1279–1284.

[73]Bezdek, J. C. (1973). Fuzzy mathematics in pattern classification. PhD thesis, Applied Math. Center, Cornell University Ithaca.

[74]Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks, 51(12), 3448–3470.

[75]Joo, D., Hong, T., & Han, I. (2003). The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors. Expert Systems with Applications, 25(1), 69–75.

[76]Endorf, C., Schultz, E., & Mellander, J. (2004). Intrusion detection and prevention. California: McGraw-Hill.

[77]Beghdad, R. (2008). Critical study of neural networks in detecting intrusions. Computers and Security, 27(5-6), 168–175.

[78]Manikopoulos, C., & Papavassiliou, S. (2002). Network intrusion and fault detection: A statistical anomaly approach. IEEE Communications Magazine, 40(10), 76–82.

[79]KDD Cup 1999 Intrusion Detection Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2010.

[80] CRFall http://en.pudn.com/downloads148/sourcecode/.../detail640513_en.html.

[81]Axelsson, S. (2003). The base-rate fallacy and the difficulty of intrusion detection. ACM Transaction on Information and System Security, 3, 186–205.

[82] Kotagiri Ramamohanarao, Kapil Kumar Gupta, Tao Peng, and Christopher Leckie. The Curse of Ease of Access to the Internet. In *Proceedings of the 3rd International Conference on Information Systems Security (ICISS)*, pages 234–249. Lecture Notes in Computer Science, Springer Verlag, Vol (4812), 2007

[83] L. T. Heberlein, G.V. Dias, K. N. Levitt, B. Mukherjee, J.Wood, and D.Wolber. A Network Security Monitor. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 296–304. IEEE, 1990.

[84] Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu. Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 31(4):266–274, 2001.

[85] S.E.Smaha. Haystack: An Intrusion Detection System. In Proceedings of the 4th Aerospace Computer Security Applications Conference, pages 37–44. IEEE, 1988.

[86] C. Sutton and A. McCallum. *Introduction to* Statistical Relational Learning: An Introduction to ConditionalRandom Fields for Relational Learning. MIT Press, 2006. .http://www.cs.umass.edu/˜mccallum/papers/crf-tutorial.pdf.

[87]T. G. Dietterich. Machine learning for sequential data: A review.In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Lecture Notes in Computer Science, Springer-Verlag, No. (2396), 2002.

[88] A. Ratnaparkhi. A maximum entropy model for part-ofspeech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133– 142. Association for Computational Linguistics, 1996.

[89] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of Eighteenth International Conference on Machine Learning, ICML*, pages 282–289, 2001.