Chapter 2

Literature Survey

2.1 Introduction

This chapter presents a review of trends and challenges in the Intrusion Detection System research. The network information systems play crucial roles for most governments, enterprises, and even individuals. Thus, the systems must remain not only up-and-running but also secure against any form of harmful actions such as misuse, abuse and attack. However, the networked information systems have historically undergone ever-increasing events of harassment by hostile parties despite the existence of a variety of security technologies. In general, security can be achieved in three phases such as prevention, detection and correction. Intrusion prevention is a pre-emptive approach to network security. It is used to identify potential threats and respond to them swiftly. It also monitors network traffic. Intrusion prevention systems also have the ability to take immediate action against the attack based on a set of rules established by the network administrator. Prevention is the most ideal solution, but unfortunately the history shows us it can’t be achieved perfectly. Even so, it is a very bad idea to entirely rely on prevention. It’s because in case an attacker somehow finds a way to make a security a good-for-nothing, the cost for fixing the vulnerability and restoring the system back to normal condition must be incredibly expensive in the later phase if there is no preparation for that. Therefore, security protection systems are better off having ready-to-go correction mechanisms as well. By well-designed correction mechanisms, compromised or malfunctioning systems can be quickly repaired and restored to normal condition. Prevention is effective before successful intrusions. Correction is active after successful intrusions. However, once successful attacks eventually manage to get through prevention, it’s a matter of time that the whole system is attacked, compromised, and malfunctioned. Thus, we need to have an interim stage such as detection phase, which is
positive during intrusion. By a detection mechanism, even if prevention fails to stop intrusion, a protected system can be at least aware of being attacked so that the system can take some actions to reduce the probability of propagating damage and loss. By doing so, detection can shorten the gap between prevention and correction for minimizing the net cost. For this reason, the detection phase of security has gradually gaining its gravity as an essential element for a complete security solution. Intrusion Detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. More specifically, the goal of intrusion detection is to identify entities attempting to subvert in-place security controls.

Client can be connected to internet using router, but for security purpose firewall can be configured between client or intranet and the edge router. Firewalls are based on filtering mechanisms specified by a set of rules, known as a policy. The basic operations of firewalls are filtering packets passing through specific network ports or hosts. In other others, firewalls prevent unauthorized traffic from passing through a protected system. However, some traffic must pass in and out of a firewall in order for the protected system to be useful. If a system is kept completely isolated, the system may be secure but not useful at all. That said, firewalls provide an all-or-nothing type of security. In other words, a firewall cannot protect against data that are allowed to pass through the firewall. Besides, firewalls cannot stop insider attacks. Security provided by the firewall is not sufficient. So IDS is installed between intranet and the firewall. Firewall will prevent the intruders to attack intranet. Even though the intruder manages to break the security of firewall, IDS residing behind firewall will detect the attacks. IDS monitor network traffic and monitors for suspicious activity and alerts the system or network administrator. The security architecture is as shown in Figure 2.1.

![Figure 2.1: Network based IDS](image-url)
In an information system, intrusions are the activities that violate security policies of the system, and intrusion detection is the process used to identify intrusions. Intrusion detection techniques have been traditionally classified as misuse or anomaly detection. IDS can be classified further based on audit source location, approach, analysis timing and behavior as shown in Figure 2.2. Each of the above type has its own advantages and disadvantages. Network-based intrusion detection system (NIDS) resides on the perimeter of a protected network to detect suspicious traffic coming from the outside network, namely the Internet. NIDS inspects and analyzes network traffic and devices at the network, transport, and application layer to detect malicious activities. It is a necessary security component because most of the attacks come from the Internet. Host-based intrusion detection system (HIDS) is installed on a particular host to monitor suspicious events occurring within the host. In other words, HIDS resides on the endpoints of the network. Unlike NIDS, HIDS monitors not only malicious network traffic but also various events occur within the protected host. The capabilities of HIDS include code analysis, buffer overflow detection, system call monitoring, application and library lists, file system monitoring, privilege misuse and abuse, system log analysis, system and application configuration analysis, and many others. The IDS can be categorized according to the methodology used as: misuse based and anomaly based IDS. The misuse type of detection works in a way that it looks for patterns of signatures or known intrusions. It is also known as Signature-based. These types of IDS have high detection rate and low false positive rate. An anomaly detection system is able to detect unknown intrusions. It works on profiles to detect unknown intrusions [23]. The profiles represent the normal behavioral activities of the users, systems, or network connections, applications. These profiles are developed by monitoring the characteristics of typical activity over a period of time. Profiles can be created based on a number of behavioral attributes like the number of emails sent by a user, the number of failed login attempts for a host, and the processor usage in a given period of time. Great care should be taken while defining profiles because currently there is no effective way to define normal profiles that can achieve high detection rate and low false positives at the same time. If the profiles are too broadly defined, some attacks might not be detected. This results in a low detection rate. On the other hand, if the profiles are too narrowly defined, some
normal activities might be detected as intrusion [24]. An active Intrusion Detection Systems (IDS) is also known as Intrusion Detection and Prevention Systems (IDPS). IDPS is configured to automatically block suspected attacks without any intervention by an operator. In response to an attack, IDPS provides real-time corrective action. The system which only monitors and analyzes network traffic activity and generate alert to potential vulnerabilities and attacks is known as passive IDS. It is not capable of performing any protective or corrective functions on its own. Another way to classify the IDS based upon the analysis timing is: period monitoring and continuous analysis.

Figure 2.2: Classification of Intrusion Detection System

2.2 Trends in Information Security
An Intrusion Detection System (IDS) requires sophisticated techniques to examine massive traffic passing through the network. The basic motivation is to measure how close a behaviour with respect to some previously established gold standard of misuse or
normal behaviour. Depending on the level of a priori or domain knowledge, it may be possible to design detectors for specific categories of attacks (e.g. DOS, R2L, U2R, and Probe). Various machine learning techniques have been proposed in an attempt to improve on the signature-based IDS. Generic machine learning approaches include clustering or data mining where the data is unlabeled. The overriding assumption is that behaviours are sufficiently different for normal and abnormal data to fall into different “clusters”. Specific examples of such algorithms include artificial immune systems [25] as well as various neural network and clustering algorithms [26]. Supervised learning algorithms are more appropriate when labelled data is available. Numbers of machine learning approaches have been proposed, including: Artificial Neural Networks (ANN), Genetic programming and Decision trees [27]. However, irrespective of the particular methodology, all such machine learning methods need to address the scalability problem. That is data sets characterizing the IDS problem are exceptionally large (by machine learning standards). The continuing evolution of the attacks also requires that any machine learning approach should have the capability for online or incremental learning. It would be useful for network administrators if machine learning solutions were transparent. Constant patterns in the evolution of information protection, network threats, and cyber-crime have appeared during last few years [28, 29]. These trends need to be taken into account in order to address the design and impact of future intrusion detection systems in a better way. Some of these trends are discussed below:

- **Average technical expertises of attackers are decreasing:** The number of security incidents has been steadily growing over the years due to the expansion of Internet. Recent attacks on the internet are severe and very sophisticated in nature. This is due to the increasing complexity of systems and the fact that more people are making use of security software [29]. However, the average technical knowledge of attackers has decreased. It is no longer necessary to be a systems expert in order to successfully launch an attack on the internet. Novice users try to break into systems by using available scripts and automated tools. These users may not know the details of how an exploit works but have the curiosity of seeing whether it actually works. These users are dangerous as they do not know the consequences of using such tools downloaded by them.
• **Numbers of attacks originated from the outside are increasing:** Years ago the rule was to see internal users abusing their privileges to compromise a local host. External attacks were not as common as today. With the connectivity provided by modern networks, more people have access to the internet and have the potential of becoming an attacker. As a result, the number of external attacks has significantly increased [30, 31, 32]. This trend also reveals an increasing number of users who try to hide through connection hopping before studying and breaking into their target [33].

• **Use of intrusion detection tools are increasing:** Three common tools usually used by security administrators are firewall, authentication system, and anti-virus software [32]. According to Computer crime & security survey 2006, intrusion detection programs are being added by more people as a component of typical corporate security suite. This survey demonstrates the importance of security monitoring and encourages the creation of better tools that can help to identify illicit activities [30].

• **Availability of automated attack tools are increasing:** Recent statistics on software vulnerabilities and security reports coincide to indicate that an increasing number of attacks on the Internet are being executed with automated attacking tools [30, 34]. The number of attack-tools available on the Internet is increasing day by day. To generate the attacks using these tools need low technical expertise of the users. These create a severe challenge to information security researchers.

• **More Distributed Denial of Service Attacks:** The percentage of denial of service attacks (DOS) are more as compare to other attacks. The number of denial of service attacks (DOS) originated from remote locations is increasing. These attacks first compromise a group of machines that will be used to launch an attack using remotely-managed tools to defeat a common target (or a few targets together). The inability of current network protocols, in particular the TCP/IP family, to preserve the integrity of packets, makes this attack most difficult one to defend [35].

### 2.3 Approaches for IDS

There are various approaches to build intrusion detection system. Some of them are statistical based and others are Machine learning based data mining approaches.
2.3.1 Statistical-Based Approaches

Statistical-based approach is one of the earliest methods used for intrusion detection. This is a frequently used method. In this approach the user or system behaviour (set of attributes) is measured by a number of variables over time. Examples of such variables are: user login, logout, number of files accessed in a period of time, usage of disk space, memory, CPU etc. The frequency of updating can vary from a few minutes to, for example, one month. The system stores mean values for each variable used for detecting if the frequency of updating exceeds that of a predefined threshold. Yet, this simple approach was unable to match a typical user behaviour model. Approaches that relied on matching individual user profiles with aggregated group variables also failed to be efficient. Therefore, a more sophisticated model of user behaviour has been developed using short- and long-term user profiles. These profiles are regularly updated to keep up with the changes in user behaviours. Statistical methods are often used in implementations of normal user behaviour profile-based Intrusion Detection Systems.

Denning proposes a statistical method for intrusion detection [2]. In that paper he discussed five types of statistical methods: operational model, mean and standard deviation model, multivariate model, Markov process model, and time series model. Li et al. utilize statistical characteristics of n-grams to detect intrusions in the host system [28].

In Statistical-Based approach [36] shows that systems with large number of input features also have a higher risk of over-fitting. However, under-fitting might occur when the capacity of the model is restricted more than required. Controlling the capacity or complexity of the models usually involves the fine tuning of a number of model-dependent (hyper) parameters. It is assumed that an intruder’s behaviour is noticeably different from that of a normal user. Statistical models are used to aggregate the user’s behaviour and distinguish an attacker from a normal user.

The Stanford Research Institute’s next-generation real-time intrusion detection expert system statistical component (NIDES/STAT) observes behaviours of subjects on a monitored computer system. It adaptively learns what is normal for individual subjects, such as users and groups [37]. The observed behaviour of a subject is flagged as a potential intrusion if it deviates significantly from the subject’s expected behaviour. The algorithm proposed by Haystack uses different statistical based algorithms, which was
adopted as the core of the host monitor in the distributed Intrusion Detection System (DIDS). Unlike NIDES/STAT, the Haystack algorithm determines resemblance to known attacks. Hence more knowledge about the possible attacks can be derived first. Then the better responses can follow the alarms. However, for these alarms the extra knowledge about possible intrusion types is required. There is a need to understand the impact of the intrusion types on the attributes of the session vectors and assign appropriate weights to these attributes to reflect the impact. In reality, the process of generating the weighted intrusion vectors is time consuming and error prone [37].

Vigna and Kemmerer use data that are sourced from network nodes, rather than the audit data, to construct profiles, for network-based intrusion detection system [38]. Some researchers propose more complex metrics and statistical models. Qu et al analyze the attacks to routing protocols by estimating the frequency of each event related to a protocol. Further, they propose a metric to describe similarity between the observed event distribution and the expected distribution [39]. Ye et al proposed a method in which it is assumed that the metric obeys the chi-square distribution to extract an event frequency vector and then measure the chi-square distance between this vector and the expected frequency vector [40]. Li and Manikopoulos propose some representative parameters of IP data flow, and they model the parameters using a hyperbolic distribution [41].

Caberera et al. assume that the first derivative of the number of observed events in a time segment obeys the Poisson distribution. From this, the Kolmogorov statistical values are extracted to measure the dissimilarity between observation network and normal behaviour signals [42]. Ye et al. represent a sequence of events in time order as a Markov stochastic process [43]. The joint probability for a particular sequence of events is used to distinguish between normal network behaviours and intrusions. In recent years, the hidden Markov model has been used in intrusion detection based on host audit data [44, 45].

**Limitations to statistical-based approach:**

- Traditionally, IDS are developed using expert knowledge of the system and attack methods. Due to the complexity of modern network system and sophistication of attackers, expert knowledge engineering is often very limited and unreliable.
• Statistics-based IDS are very sensitive to the data representation. For instance, if the representation contains irrelevant information these IDS may fail to generalize an unseen data.

• Performance of statistics-based approach depends upon the determination of appropriate threshold for detection. If the threshold is set too low, anomalous activities that are normal are flagged as intrusive resulting in high false positive rate. If the threshold is set too high, anomalous activities which are intrusive are flagged as normal resulting in high false negative rate [46].

2.3.2 Machine Learning based Data Mining Approaches

Machine learning is the study of computer algorithms that improve automatically through experience. A major focus of machine learning research is to automatically learn to recognize complex patterns and to make intelligent decisions based on data. The notion of finding useful patterns in data has been given a variety of names including data mining, knowledge discovery in databases, information harvesting, data archaeology, and data pattern analysis [47, 48]. Data Mining is the process of automatically searching large volumes of data for patterns using association rules. Data Mining can be differentiated by their different model functions and representation, preference criterion and algorithms.

Time-Based Inductive Machine

A time-based inductive machine (TIM) is used to capture a user’s behaviour pattern [49]. As a general-purpose tool, TIM discovers temporal sequential patterns in a sequence of events. The temporal sequential patterns, which are represented in the form of rules, are generated and modified from the input data using a logical inference called inductive generalization. When applied to intrusion detection, the rules describe the behaviour patterns of either a user or a group of users based on audit data. Each rule describes a sequential event pattern that predicts the next event from a given sequence of events. An example of a simplified rule produced in TIM is

\[ E_1 \rightarrow E_2 \rightarrow E_3 \Rightarrow (E_4 = 90\%, \quad E_5 = 10\%) \]  

(2.1)

Where E1, E2, E3, E4, and E5 are security events
This rule says that if $E_1$ is followed by $E_2$, and $E_2$ is followed by $E_3$, then there is a 90% chance (based on the previous observation) that $E_4$ will follow, and a 10% chance that $E_5$ will follow. TIM can produce more generalized rules than the above. For example, it may produce a rule in the form

$$E_1 \rightarrow * \Rightarrow (E_2 = 100\%)$$

(2.2)

Where as an asterisk matches any single event. Any number of asterisks is allowed in a rule. The limitation of TIM is that it only considers the immediately following relationship between the observed events. However, a user may perform multiple tasks at the same time. For example, a user may check his/her e-mail during the editing of a document. The events involved in one application, which tend to have strong patterns embedded in the sequence of events, may be interleaved with events from other applications. TIM may not be suitable for capturing the behaviour patterns of such entities as programs that usually focus on single tasks.

**Instance Based Learning**

Instance based learning (IBL) is applied to learn subjects (e.g., users) normal behaviour from temporal sequence data. IBL represents a concept of interest with a set of instances that exemplify the concept. The set of instances is called the instance dictionary. IBL requires a notion of “distance” between the instances so that the similarity of different instances can be measured and used to classify the instances.

Lane and Brodley presented an approach to the anomaly detection problem based on IBL techniques [11]. First, they transformed the observed sequential data into fixed-length vectors (called feature vectors). Specifically, they segmented a sequence of events (e.g., a sequence of user commands) into all possible overlapping sequences of length $l$, where $l$ is an empirical parameter. (Thus, each event is considered the starting point of a feature vector, and each event is replicated $l$ times.) Second, they defined a similarity measure between the feature vectors. For a length $l$, the similarity between feature vectors $X = (x_0, x_1, \ldots, x_{l-1})$ and $Y = (y_0, y_1, \ldots, y_{l-1})$ is defined by the functions eq. 2.3 and eq. 2.4 [11] below.
\[ w(X, Y, i) = \begin{cases} 
0 & \text{if } i < 0 \text{ or } x_i, y_i \\
1 + w(X, Y, i - 1) & \text{if } x_i = y_i
\end{cases} \] (2.3)

where \( w(X, Y, i) = 0 \) for \( i < 0 \) so that \( w(X, Y, 0) \) is well defined when \( x_0 = y_0 \) and

\[
\text{Sim}(X, Y) = \sum_{i=0}^{l-1} w(X, Y, i) \tag{2.4}
\]

The converse measure, distance, is defined as \( \text{Dist}(X, Y) = \text{Sim}_{\text{max}} - \text{Sim}(X, Y) \), where \( \text{Sim}_{\text{max}} = \text{Sim}(X, X) \). Intuitively, the function \( w(X, Y, i) \) accumulates weights from the most recently consecutively matched subsequences between \( X \) and \( Y \) at position \( i \), whereas \( \text{Sim}(X, Y) \) is the integral of total weights. A user profile is built to contain a collection of sequences, \( D \), selected from a user’s observed actions (e.g., commands). The similarity between the profile and a newly observed sequence, \( X \), is defined as \( \text{Sim}(D(X)) = \max YOED\{\text{Sim}(Y, X)\} \). That is, the similarity between \( X \) and \( D \) is defined as the similarity between \( X \) and a vector in \( D \) that is most similar to \( X \). Then a threshold \( r \) is chosen. If the similarity between an observed sequence \( X \) and the profile \( D \) is greater than \( r \), \( X \) is considered normal; otherwise, \( X \) is abnormal.

To reduce the storage required by the profile, they used the least-recently-used pruning strategy to keep the profile at a manageable size. As new instances are acquired and classification is performed, the profile instance selected as most similar is time stamped. In addition, they applied a clustering technique to group the instances in the profile, and they used a representative instance for each cluster [11].

This attempt shares a problem similar to that of TIM, that is, it tries to find patterns from sequences of consecutive events. As the authors have noted, a user may interrupt his/her normal work (e.g., programming) and do something different (e.g., answer an urgent email) and thus yield a different sequence of actions from his/her profile. Their solution is to use a time average of the similarity signals; however, such a solution may make real anomalies unnoticeable. In addition, the least-recently-used pruning strategy gives an attacker a chance to train the profile slowly, so that intrusive activities are considered normal ones [11].
Audit Data Analysis and Mining

Audit data analysis and mining (ADAM) proposes applying data mining techniques to discover abnormal patterns in large amounts of audit data [50, 51]. Using data mining techniques, ADAM has the potential to provide a flexible representation of the network traffic pattern. It also, uncover some unknown patterns of attacks that cannot be detected by other techniques, and accommodate the large amount of network audit data that keeps growing in size.

ADAM uses several data-mining-related techniques to help detect abnormal network activities. The first technique ADAM uses is inspired by association rules. Given a set I of items, an association rule is a rule of the form $X \rightarrow Y$, where X and Y are subsets (called item sets) of I and $X \cap Y = \emptyset$. Association rules are usually discovered from a set $T$ of transactions, where each transaction is a subset of I. The rule $X \rightarrow Y$ has a support $s$ in the transaction set $T$ if $s\%$ of the transactions in $T$ contain $X \cup Y$, and it has a confidence $c$ if $c\%$ of the transactions in $T$ that contain $X$ also contain $Y$.

However, ADAM doesn’t use association rules directly; instead, it adopts the item sets that have large enough support (called large item sets) to represent the pattern of network traffic.

The second technique ADAM uses is called domain-level mining. Intuitively, it tries to generalize the event attribute values used to describe a network event. For example, an IP address that belongs to the subnet ise.gmu.edu can be generalized to ise.gmu.edu, gmu.edu and edu. An advantage of this approach is that it provides a way to aggregate the events that share some commonality and may discover more attacks.

The third technique ADAM uses is classification. ADAM is innovative in that classification is used to classify the output of the mining of large item sets. Classification algorithms such as C4.5 decision tree, naive Bayes, cascading classifier (which uses decision tree followed by naive Bayes and vice versa), and inductive rule learner have been studied. The results show that classification is quite effective in reducing false alarms.

Finally, ADAM uses the pseudo-Bayes estimator to accommodate unknown attacks. The training data is represented as a set of vectors, each of which corresponds to an event and is labelled as normal or as a known class of attacks. An additional class is then
considered to represent the unknown attacks. Because the unknown attacks haven’t been observed in the training data, the probability $P(x|\text{class} = \text{unknown})$, where $x$ is a training vector, is zero. The pseudo-Bayes estimator is used to smooth all the conditional probabilities $P(x|\text{class})$ so that $P(x|\text{class} = \text{unknown})$ is assigned a (small) probability. These conditional probabilities are then used to build a naive Bayes classifier.

The limitation of ADAM is that it can detect an attack only when it involves a relatively large number of events during a short period of time. This limitation occurs because ADAM raises an alarm only when the support of an unexpected rule (i.e., association of event attributes) exceeds a threshold. A potential source of confusion is that different data mining techniques assume different input data representations. For example, association rules have historically been discussed under the assumption that the input data is represented as a set of transactions [134, 135]. Data mining technique is used to construct rules describing normal network behaviours [52, 53]. The rules include association rules that describe frequency associations between any two fields of the network record database. Deviations from these rules indicate an attack on the network.

Han et al. analyze the content for network data packages and use the data-mining techniques to acquire attack signatures [53].

**Specification-Based Methods**

A specification-based approach for intrusion detection is proposed by Ko et al [54]. The idea is to use traces, ordered sequences of execution events, to specify the intended behaviours of concurrent programs in distributed system. A specification describes valid operation sequences of the execution of one or more programs, collectively called a (monitored) subject. The drawback of this approach is that substantial work is required to specify accurately the behaviour of the many privileged system programs, and these specifications will be operating-system specific.

**Computer Immunological Approach**

The computer immunological approach is based on an analogy of the immune system’s capability of distinguishing self from non-self [55]. This approach represents self as a collection of strings of length $l$, where $l$ is a system-wide parameter. A string of length $l$ is considered non-self if it does not match any string belonging to self. Forrest et al proposed “r-contiguous-bits” matching rule to distinguish self from nonself: two $l$-bit
strings match each other if they are identical in at least $r$ contiguous positions. As a result, detectors can be generated more efficiently for this particular matching rule.

Hofmeyr et al proposed architecture for an artificial immune system for anomaly detection using short sequences of system calls [55]. For an observed sequence of system calls, this approach extracts all subsequences of length $l$ and computes the distance $d_{\text{min}}(i)$ between each subsequence $i$ and the normal database as $d_{\text{min}}(i) = \min\{d(i,j)\}$ for all sequences $j$ in the normal database, where $d(i,j)$ is the Hamming distance between sequences $i$ and $j$ (i.e., the number of different bits in sequences $i$ and $j$). The anomaly score of the observed sequence of system calls is then the maximum $d_{\text{min}}(i)$ normalized by dividing the length of the sequence. This approach raises an alarm if the anomaly score is above a certain threshold.

The advantage of this approach is that it has a high probability of detecting anomalies using a small set of self-strings based on short sequences of system calls. In addition, it does not require any prior knowledge about attacks. The disadvantage is that it requires a complete set of self-strings in order not to mistake self for non-self. This requirement may be trivial for such applications as virus detection, but it is very difficult for intrusion detection, where some of normal behaviours cannot be foreseen when the detectors are being generated.

2.4 Learning Methods

Machine learning and pattern recognition methods have been utilized to detect intrusions. Learning algorithms can be categorized as unsupervised, supervised and semi-supervised. This section gives the review of all these learning based methods.

2.4.1 Unsupervised Learning

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns. It learns from unlabeled examples [56]. The objective of unsupervised learning may be to cluster examples together on the basis of their similarity. Unsupervised learning methods can detect new kinds of attacks provided they exhibit unusual character in some feature space. Unsupervised learners are not provided with classifications. In
fact, the basic task of unsupervised learning is to develop classification labels automatically. Unsupervised algorithms seek out similarity between pieces of data in order to determine whether they can be characterized as forming a group. These groups are termed clusters, and there are whole families of clustering machine learning techniques. Examples of unsupervised learning for intrusion detection include K-means-based approaches and self-organizing feature map (SOM)-based approaches [57].

1. **K-means**

Guan et al. propose a K-means-based clustering algorithm, which is named as Y-means, for intrusion detection [58]. Xian et al. combine the fuzzy K-means method and a clonal selection algorithm to detect intrusions. Jiang et al. use the incremental clustering algorithm that is an extension of the K-means algorithm to detect intrusions [59].

2. **Self Organizing Map (SOM)**

SOM is based on competitive learning. The Winner takes all neuron and forms a topographic map of input patterns i.e. spatial locations of neurons in the lattice are indicative of statistical features contained in the input patterns. It is a case of supervised learning and typically used for multilayer perceptrons. There are two stages, forward pass and backward pass. In forward pass input signal propagates forward to produce the output. In backward pass, synaptic weights are updated in accordance with the error signal, which is then propagated backwards.

Kayacik et al. proposed a hierarchical SOM approach for intrusion detection. In this work specific attention is given to the hierarchical development of abstractions, which is sufficient to permit direct labelling of SOM nodes with connection type [26]. Hoglund et al. extract features that describe network behaviours from audit data, and they use the SOM to detect intrusions [60]. Sarasamma et al. proposed a hierarchical SOM which uses the classification capability of the SOM on selected dimensions of the data set to detect anomalies. Their results are among the best known for intrusion detection [61].

2.4.2 **Supervised Learning**

Supervised machine learning methods require labelled data for training. The objective of supervised learning is to learn about assigning correct labels to new unseen examples of the same task. For supervised learning for intrusion detection, there are mainly neural
network (NN)-based approaches [62, 63], and support vector machine (SVM) based approaches [64, 65].

1. **Neural Network (NN)**

A neural network based IDS typically consist of a single neural network based on either misuse detection or anomaly detection. Misuse detectors analyze system activities and try to find a match between these activities and known attacks having definitions or signatures introduced to the system beforehand. Anomaly detectors detect behaviours on a computer or computer network that are not normal. [66]. Neural network with good pattern classification abilities used for misuse detection, such as Multilayer Perceptron, Radial Basis function networks, etc.

Bonifacio et al. proposed a NN to distinguish between intrusions and normal behaviours. They unify the coding of categorical fields and character string fields in order to map the network data to NN [67]. Rapaka et al. used execution numbers of system calls in a host machine as the features of network behaviours to train the NN [68]. Zhang et al. proposed an approach for intrusion detection using hierarchical NNs [63]. Han et al used evolutionary NNs to detect intrusions [69].

2. **Support Vector Machine (SVM)**

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

Mukkamala et al used SVMs to distinguish between normal network behaviours and intrusions and further identify important features for intrusion detection [70]. Mill et al proposed the TreeSVM and ArraySVM for solving the problem of inefficiency of the sequential minimal optimization algorithm for the large set of training data in intrusion detection [71]. Zhang et al proposed an approach for online training of SVMs for real-time intrusion detection based on an improved text categorization model [65].
Wang et al. used one class SVM based on one set of examples belonging to a particular class and no negative examples rather than using positive and negative examples. Neither of these approaches addresses the reduction of the training time of SVM, which prohibits real-time usage of these approaches [72].

Apart from the above mentioned supervised-learning-based approaches for intrusion detection, decision tree and discriminate analysis are applied to detect intrusions.

2.4.3 Semi-supervised learning

With the immense amount of network and host data available, expert labelling of the data is very expensive. The labelled data available is often from controlled environments. For instance, the 1998 and 1999 intrusion detection evaluations from DARPA/MIT Lincoln lab [20] have the ground truth information, but the data itself have been shown to be not representative of real environments. This proves to be a bottleneck in applying supervised learning methods to detect novel or unknown attacks. Hence, relying only on supervised learning methods which requires a large amount of labelled data is impractical for real network environment. This motivates a need for a new and more practical learning framework. Semi-supervised learning methods can leverage unlabeled examples in addition to labelled ones. Semi-supervised learning methods received significant attention, and are more suitable for real network environment because these methods require a small quantity of labelled data while still taking advantage of the large quantities of unlabeled data.

Several semi-supervised classification algorithms that use unlabeled data to improve classification accuracy have become popular in the past few years. It includes co-training [73], transductive support vector machines [74], and using Expectation Maximization to incorporate unlabeled data into training [75, 76]. Unlabeled data have also been used to learn good distance measures in the classification setting [77]. Machine learning communities and researchers have proposed techniques based on Markov chains [78], hidden Markov models, association rules. Forrest et al. have developed IDSs based on models of biological immune systems [79]. These artificial immune systems employ instance- or rule-based learners in either anomaly detection or misuse detection frameworks effectively. A good review of semi-supervised classification methods is given by Matthias Seeger and Olivier Chapelle et al [80, 81]. Negar et al proposed the
new framework to rebuilds the classifier model using selected data from test data set to improve the accuracy of the model. [82].

Brief introduction of the semi-supervised methods given below:

**Generative Models**

Generative models are the oldest semi-supervised learning method. It assumes a model $p(x, y) = p(y)p(x|y)$ where $p(x|y)$ is an identifiable mixture distribution. A Gaussian mixture model is an example of this model. With large amount of unlabeled data, the mixture components can be identified; then there is a need to identify one labelled example per component to fully determine the mixture distribution. One can think of the mixture components as ‘soft clusters’. While using this model following things has to be considered.

a) Identifiability: The mixture model ideally should be identifiable

b) Model Correctness: If the mixture model assumption is correct, unlabeled data is guaranteed to improve accuracy is get improved when mixture model assumption is correct and if the model is wrong then it is lowered by unlabeled data.

c) EM Local Maxima: Exception Maximization is prone to local maxima. If a local maximum is far from the global maximum, unlabeled data causes to lowered accuracy.

d) Cluster and label: Instead of using a probabilistic generative mixture model, some approaches uses various clustering algorithms to cluster the whole data set, then label each cluster with labelled data. Sometimes these approaches are hard to analyze due to their algorithm nature.

Nigam et al. applied EM algorithm on mixture of multinomial for the task of text classification [83]. L. Baluja used the same algorithm on a face orientation discrimination task [84]. Fujino et al. extended generative mixture models by including a bias correction term and discriminative training using the maximum entropy principle [85]

**Self-Training**

In self training a classifier is first trained with the small amount of labelled data. The classifier is then used to classify the unlabeled data. The most confident unlabeled data, together with their predicted labels, are added to the training data set. The classifier is re-trained and the procedure repeated. Classifier uses its own predictions to teach itself.
Self-training has been applied to several natural language processing tasks. For word sense Yarowsky used self-training [86]. Riloff et al. used it to identify subjective nouns [87]. Self-training has also been applied to parsing and machine translation. Maeireizo et al. classify dialogues as ‘emotional’ or ‘non-emotional’ with a procedure involving two classifiers. Self-training has also been applied to parsing and machine translation [88]. Rosenberg et al. applied self-training to object detection systems from images, and show the semi-supervised technique compares favourably with a state of-the-art detector [89].

**Co-Training**

Co-training assumes that two or possibly more learners are each trained on a set of examples, but with each learner using a different, and ideally independent, set of features for each example. In co-training, unlabeled data helps by reducing the version space size. In other words, the two classifiers (or hypotheses) must agree on the much larger unlabeled data as well as the labelled data.

Nigam et al performed extensive empirical experiments to compare co-training with generative mixture models and EM. They have proved that co-training performs well if the conditional independence assumption indeed holds [83]. W. Wang et al presented a new analysis on the co-training. According to their analysis the co-training process is viewed as combinative label propagation over two views; this provides a possibility to bring the graph-based and disagreement based semi-supervised methods into a unified framework [90].

**Multiview Learning**

More generally, learning paradigms can be defined that utilize the agreement among different learners. The particular assumptions of Co-Training are in general not required by multiview learning models. Instead, multiple hypotheses (with different inductive biases, e.g., decision trees, SVMs, etc.) are trained from the same labelled data set, and are required to make similar predictions on any given unlabeled data. Sindhwani et al. and Brefeld et al applied this method to semi-supervised regression and the more challenging structured output spaces [91, 92, 93]. Some theoretical analysis on the value of agreement among multiple learners can be found in [94, 95].
**Transductive SVMs (S3VMs)**

Discriminative methods work on \( p(y|x) \) directly. This brings up the danger of leaving \( p(x) \) outside of the parameter estimation loop, if \( p(x) \) and \( p(y|x) \) do not share parameters. Notice \( p(x) \) is usually got from unlabeled data. It is believed that if \( p(x) \) and \( p(y|x) \) do not share parameters, semi-supervised learning cannot help [80]. Transductive support vector machines (TSVMs) build the connection between \( p(x) \) and the discriminative decision boundary by not putting the boundary in high density regions. TSVM is an extension of standard support vector machines with unlabeled data. In a standard SVM only the labelled data is used, and the goal is to find a maximum margin linear boundary in the Reproducing Kernel Hilbert Space. In a TSVM the unlabeled data is also used. The goal is to find a labelling of the unlabeled data, so that a linear boundary has the maximum margin on both the original labelled data and the (now labelled) unlabeled data. The decision boundary has the smallest generalization error bound on unlabeled data [36]. Intuitively, unlabeled data guides the linear boundary away from dense regions. Zhang et al pointed out that TSVMs may not behave well under some circumstances [24]. The maximum entropy discrimination approach also maximizes the margin, and is able to take into account unlabeled data, with SVM as a special case [96].

**Gaussian Processes**

Lawrence et al. proposed a Gaussian process approach which can be viewed as parallel to TSVM. The key difference to a standard Gaussian process is in the noise model. A ‘null category noise model’ maps the hidden continuous variable \( f \) to three instead of two labels, specifically to the never used label ‘0’ when \( f \) is around zero. On top of that, it is restricted that unlabeled data points cannot take the label 0. This pushes the posterior of \( f \) away from zero for the unlabeled points. It achieves the similar effect of TSVM where the margin avoids dense unlabeled data region [97]. However nothing special is done on the process model. Therefore all the benefit of unlabeled data comes from the noise model. A very similar noise model is proposed in [56] for ordinal regression.

**Information Regularization**

Szummer et al introduced the original information regularization principle in a study of the influence of the distribution of the objects on the variations in their label, when the objects are represented by a one dimensional real number [98].
Entropy Minimization

Zhu suggest the hyper parameter learning method which uses entropy minimization [99]. Grandvalet et al used the label entropy on unlabeled data as a regularizer. By minimizing the entropy, the method assumes a prior which prefers minimal class overlap [100].

Graph-Based Methods

Graph-based semi-supervised methods define a graph where the data (both labelled and unlabeled) is represented by vertices in a graph. Graph edges link vertices that are likely to have the same label. Edge weights govern how strongly the labels of the nodes linked by the edge should agree [101].

Regularization by Graph

Many graph-based methods can be viewed as estimating a function $f$ on the graph. One wants $f$ to satisfy two things at the same time: 1) It should be close to the given labels $y_L$ on the labelled nodes, and 2) It should be smooth on the whole graph. This can be expressed in a regularization framework where the first term is a loss function, and the second term is a regularizer. Several graph-based methods listed here are similar to each other. They differ in the particular choice of the loss function and the regularizer [101].

Mincut

Blum et al pose semi-supervised learning as a graph mincut (also known as st-cut) problem. In the binary case, positive labels act as sources and negative labels act as sinks. The objective is to find a minimum set of edges whose removal blocks all flow from the sources to the sinks [102].

Discrete Markov Random Fields: Boltzmann Machines

The proper but hard way is to compute the marginal probabilities of the discrete Markov random fields. This is inherently a difficult inference problem. Zhu et al attempted exactly this, but were limited by the MCMC sampling techniques (they used global Metropolis and Swendsen-Wang sampling) [103].

Gaussian Random Fields and Harmonic Functions

The Gaussian random fields and harmonic function methods in is a continuous relaxation to the difficulty discrete Markov random fields (or Boltzmann machines) [104]. It can be viewed as having a quadratic loss function with infinity weight, so that the labelled data are clamped (fixed at given label values).
Local and Global Consistency

The local and global consistency method [105] uses the loss function \[ \sum_{i=1}^{n} (f_i - y_i)^2 \]
and the normalized Laplacian \( D^{-1/2} \Delta D^{-1/2} = I - D^{-1/2} WD^{-1/2} \) in the regularizer,
\[
\frac{1}{2} \sum_{ij} w_{ij} (f_i/\sqrt{D_{ii}} - f_j/\sqrt{D_{jj}})^2 = f^T D^{-1/2} \Delta D^{-1/2} f
\] (2.5)

Where, \( W \) is an affinity matrix, \( D \) is a diagonal matrix with its \((i, i)\) element equal to the sum of the \(i\)th row of \( W \), \( f \) is an indicator function of the cut.

Tikhonov Regularization:

The Tikhonov regularization algorithm in [106] uses the loss function and regularizer:
\[
\frac{1}{k} \sum_{i} (f_i - y_i)^2 + \gamma f^T S f
\] (2.6)

Where \( S = \Delta \) or \( \Delta^p \) for some integer \( p \).

Manifold Regularization:

The manifold regularization framework [107] employs two regularization terms:
\[
\frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \| f \|_K^2 + \gamma_I \| f \|_I^2
\] (2.7)

where \( V \) is an arbitrary loss function, \( K \) is a ‘base kernel’, e.g. a linear or Radial basis function (RBF) kernel. \( i \) is a regularization term induced by the labelled and unlabeled data. Sindhwani et al. give a semi-supervised kernel that is not limited to the unlabeled points, but defined over all input space. The kernel thus supports induction [108].

Graph Kernels from the Spectrum of Laplacian:

For kernel methods, the regularizer is a (typically monotonically increasing) function of the Reproducing Kernel Hilbert Spaces (RKHS) norm \( \| f \|_K = f^T K^{-1} f \) with kernel \( K \). Such kernels are derived from the graph, e.g. the Laplacian.

Spectral Graph Transducer

The spectral graph transducer can be viewed with a loss function and
\[
\text{Min } c(f - \gamma)^T C(f - \gamma) + f^T L f \text{ regularizer s.t. } f^T 1 = 0 \text{ and } f^T f = n
\]

Where \( \gamma = \sqrt{l_-/l_+} \) for positive labelled data and \( -\sqrt{l_-/l_+} \) for negative labelled data.

\( L \) can be the combinatorial or normalized graph Laplacian, with a transformed spectrum. \( c \) is a weighting factor, and \( C \) is a diagonal matrix for misclassification costs. Pham et al
perform empirical experiments on word sense disambiguation, comparing variants of co-
training and spectral graph transducer. The authors notice spectral graph transducer with
carefully constructed graphs (‘SGTCotraining’) produces good results [109].

**Local Learning Regularization**

The solution of graph-based methods can often be viewed as local averaging. For
example, the harmonic function solution if we use an un-normalized Laplacian satisfies
the averaging property:

\[ f(x_i) = \frac{\sum_{j \sim i} w_{ij} f(x_j)}{\sum_{j \sim i} w_{ij}} \]  \hspace{1cm} (2.8)

In other words, the solution \( f(x_i) \) at an unlabeled point \( x_i \) is the weighted average of its
neighbours’ solutions. Note the neighbours are usually unlabeled points too, so this is a
self-consistent property.

**Tree-Based Bayes**

Kemp et al. define a probabilistic distribution \( P(Y | T) \) on discrete (e.g. 0 and 1) labelling
\( Y \) over an evolutionary tree \( T \). The tree \( T \) is constructed with the labelled and unlabeled
data being the leaf nodes. The labelled data is clamped. The authors assume a mutation
process, where a label at the root propagates down to the leaves. The label mutates with a
constant rate as it moves down along the edges. As a result the tree \( T \) (its structure and
edge lengths) uniquely defines the label prior \( P(Y | T) \). Under the prior if two leaf nodes
are closer in the tree, they have a higher probability of sharing the same label. One can
also integrate over all tree structures [110].

Szummer et al. perform a t-step Markov random walk on the graph. The influence of one
example to another example is proportional to how easy the random walk goes from one
to the other [111]. It has certain resemblance to the diffusion kernel. The parameter \( t \) is
important. Chapelle et al. use a density-sensitive connectivity distance between nodes \( i, j \)
(a given path between \( i, j \) consists of several segments, one of them is the longest; now
consider all paths between \( i, j \) and find the shortest ‘longest segment’) [112].
Exponentiating the negative distance gives a graph kernel. Bousquet et al. propose
‘measure-based regularization’, the continuous counterpart of graph-based regularization.
The intuition is that two points are similar if they are connected by high density regions.
They define regularization based on a known density \( p(x) \) and provide interesting theoretical analysis [113].

### 2.5 Related Work

The goal of intrusion detection is to monitor network assets to detect anomalous behaviour and misuse. Pattern matching algorithms solve the general pattern matching problem for intrusion detection. That is, given a fixed and finite non-empty set of patterns as an input string, they find all occurrences of any of the patterns in the input string [114]. In this problem the input string is finite as well, but often a set of (multiple) input strings is used as input when searching for the patterns. In this work, the input strings will be packets in the detection engine of the NIDS, and therefore, there will be many of them to process rapidly. The general pattern matching problem has a specific instance that has been shown empirically as easier to solve than the general problem. In 1980, the work started on monitoring threats. In 1997, Dorothy proposed the concept of intrusion detection as a solution to the problem of providing a sense of security in computer systems. This model is rule-based pattern matching system. Statistical approaches compare the recent behaviour of a user of a computer system with observed behaviour and any significant deviation is considered as intrusion. This approach requires construction of a model for normal user behaviour.

Perhaps the most basic requirement for general purpose pattern matching is an ability to match multiple patterns quickly to the point where it effectively happens simultaneously. This is due to the fact that pattern matching in NIDS typically requires matching a very large number of patterns. When presented with the task of intrusion detection, it is observed that the number of known intrusions is growing and is almost surely to continue to do so. This growth was observed in the past in the rapid expansion of the size of the signature database for the Snort NIDS. Tuck et al. display a graph of the growth of the Snort rule database, indicating the database nearly tripled between the years of 2001 and 2004 [115]. Also a single rule may contain multiple signatures (patterns to search for). However, often many signatures are disabled in tailoring to the individual deployment needs, resulting in the opposite effect. Furthermore, depending on the NIDS there may be multiple pattern sets used, depending on the particular input. Generally if the pattern set
size increases there is decrease in the performance of pattern matching algorithms, and different algorithms scale differently. However, it is usually the pattern matching algorithm of the pattern matching engine is crucial to NIDS performance.

Zhou Chunyue et al presented the design and implementation of a pattern matching NIDS. The pattern matching based NIDS consists of four modules: Collection Module, Analyze Module, Response Module and Attack Rule Library. The system was based on CIDF architecture. The proposed system has improved the performance of detection engines due to an improved algorithm based on the current BM algorithm [116].

A novel Intrusion Detection approach was presented by Chi-Ho Tsang et al that extracted accurate and interpretable fuzzy rule-based knowledge from network traffic data using an agent-based evolutionary framework. They presented a multi-objective genetic fuzzy system for anomaly intrusion detection [117]. The experimental results on KDD-Cup99 intrusion detection benchmark data demonstrate that system can achieve high detection rate for intrusion attacks and low false positive rate for normal network traffic. A model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse was described by [2]. The model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model includes profiles for representing the behaviour of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behaviour from audit records and for detecting anomalous behaviour. The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system. Intrusion detection is a critical component of secure information systems.

S. Mukkamala et al. addressed the issue of identifying important input features in building an Intrusion Detection System (IDS). Since elimination of the insignificant and/or useless inputs leads to a simplification of the problem, faster and more accurate detection may result. Feature ranking and selection, therefore, is an important issue in intrusion detection. The important aspect was to identify key intrusion detection features that aid in achieving faster detection (real time detection) and higher accuracy rate (low
false alarm rate). They ranked the input features using neural networks by analyzing the detection accuracy, false positive rate and false negative rate [70].

Mehdi Moradi et al. presented a neural network approach to intrusion detection. A Multi Layer Perceptron (MLP) is used for intrusion detection based on an on-line analysis approach. While most of the previous studies have focused on classification of records in one of the two general classes - normal and attack, this research aimed to solve a multi class problem in which the type of attack is also detected by the neural network. Different neural network structures are analyzed to find the optimal neural network with regards to the number of hidden layers. An early stopping validation method is also applied in the training phase to increase the generalization capability of the neural network. The results show that the designed system is capable of classifying records with about 91% accuracy with two hidden layers of neurons in the neural network and 87% accuracy with one hidden layer [118].

Wenke Lee et al. uses data mining algorithms to compute activity patterns from system audit data and extracts predictive features from the patterns. It then applies machine learning algorithms to the audit records that are processed according to the feature definitions to generate intrusion detection rules [119].

There are many approaches to recognize patterns that involve using finite automata (also referred to as finite state machines). The Aho-Corasick (AC) algorithm is one such classic algorithm [120]. AC algorithm also shares characteristics with the Knuth-Morris-Pratt algorithm [121, 122]. The idea is that a finite automaton is constructed using the set of keywords during the pre-computation phase of the algorithm and the matching involves the automaton scanning the input text string reading every character in y exactly once and taking constant time for each read of a character.

Commentz-Walter achieved creating a mesh of both the Aho-Corasick multiple keyword pattern matching algorithm which has a running time linear in n and the Boyer-Moore single-keyword pattern matching algorithm which runs in time sub-linear in n on average [123]. The resulting multiple-keyword pattern matching algorithm matches multiple patterns simultaneously using a tree-like structure similar to Aho and Corasick, and using skips or shifts (i.e. filtering) similar to Boyer and Moore. Commentz-Walter also noted
that the quadratic (O(nm)) worst-case running time behaviour of the Boyer-Moore algorithm could be improved upon to be linear in n. [124].

Ching-Hao Mao et al used Co-training and Active Learning based Approach for Multi-view intrusion detection for semi-supervised approach [125]. Chien-Yi Chiu et al proposed Semi-supervised Learning for False Alarm Reduction. They use Feature selection using information gain and gain ratio and Over-sampling positive points before base learner training the classifier [126]. Jimin Li et.al proposes a novel Semi-supervised SVM Based on Tri-training for Intrusion Detection. They use three different SVMs as the classification algorithm. They use UCI data sets and application to the intrusion anomaly detection show that tri-training can improve the classification accuracy of SVM and its improved algorithms [127].

2.6 KDDCup99 Data set

The data used for this work comes from the KDD Cup competition data set [20]. This data is part of the data collected from the MIT Lincoln Labs 1998 DARPA Intrusion Detection Evaluation Program and is considered benchmark data for evaluating intrusion detection systems. Attack taxonomies and the resulting classifications are of interest to us because we envisage using a classification of attacks to identify the input to the evaluation of IDSs. Computer security attacks and vulnerabilities have been classified in many ways; however, so far no commonly accepted reference classification exists.

In the 1998 DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN. The LAN was operated like a real environment, but being blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. Of this database a subset of 494021 data were used, of which 20% represent normal patterns.

This classification distinguishes four classes of attacks

A. DOS: denial of service
B. R2L: unauthorized access from a remote machine
C. U2R: unauthorized access to local super user (root) privileges
D. Probing: surveillance and other probing
Figure 2.3 shows the detail break down of these 4 attacks into 22 attacks.

Figure 2.3: Attack Breakdown

A. Denial of Service Attacks
Denial of service attack is a class of attacks in which an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Back, Land, Neptune, Ping of death, Smurf, Teardrop.

B. User to Root Attacks
User to root exploits are a class of attacks in which an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Examples are buffer overflow, loadmodule, perl, rootkit.
C. Remote to User Attacks
A remote to user attack is a class of attacks in which an attacker sends packets to a machine over a network but who does not have an account on that machine; exploits some vulnerability to gain local access as a user of that machine. Examples are ftp_write, guess passwd, imap, phf, warezclient, warezmaster, multihop, spy.

D. Probing
Probing is a class of attacks in which an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are ipsweep, nmap, satan, portsweep.

2.7 Some Current Intrusion Detection Systems

**Beltane:** It is a web-based central management console for the Samhain file integrity / intrusion detection system. It enables the administrator to browse client messages, acknowledge them, and update centrally stored file signature databases. As the Samhain daemon keeps a memory of file changes, the file signature database need only be up to date when the daemon restarts and downloads the database from the central server. Beltane allows you to use the information logged by the client in order to update the signature database

**OSSEC:** It is an Open Source Host-based Intrusion Detection System. It performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response.

**Samhain:** It is Host-based Intrusion Detection System(HIDS) provides file integrity checking and log file monitoring/analysis, as well as rootkit detection, port monitoring, detection of rogue SUID executables, and hidden processes.

**Sectool:** It is a security tool that can be used both as a security audit as well as a part of an intrusion detection system. It consists of set of tests, library and textual/graphical frontend. Tests are sorted into groups and security levels. Administrators can run selected tests, groups or whole security levels.

**Snort:** Snort is one of the best-known lightweight IDSs, which focuses on performance, flexibility, and simplicity. It is an open-source Intrusion Detection System that is now in
quite widespread use. Snort is a network based IDS which employs signature based
detection methods. It can detect various attacks and Probes including instances of buffer
overflows, stealth port scans, common gateway interface attacks, and service message
block system Probes [4].

**Tripwire:** When an attack takes place, attackers usually replace critical system files with
their versions to inflict damage. Tripwire is an open-source host-based tool, which
performs periodic checks to determine which files are modified in the file system. To do
so, Tripwire takes snapshots of critical files. Snapshot is a unique mathematical signature
of the file where even the smallest change results in a different snapshot. If the file is
modified, the new snapshot will be different than the old one; therefore critical file
modification would be detected. Tripwire is different from the other intrusion detection
systems because rather than looking for signs of intrusion, Tripwire looks for file
modifications [128].

**Psad:** It is a collection of three lightweight system daemons (two main daemons and one
helper daemon) that run on Linux machines and analyze IP tables log messages to detect
port scans and other suspicious traffic. Psad incorporates many signatures from the Snort
Intrusion Detection System to detect probes for various backdoor programs (e.g.
EvilFTP, GirlFriend, SubSeven), DDOS tools (mstream, shaft), and advanced port scans
(FIN, NULL, XMAS) which are easily leveraged against a machine via nmap. When
combined with fwsnort and the Netfilter string match extension, psad is capable of
detecting many attacks described in the Snort rule set that involve application layer data.

**Smooth-Sec:** It is a ready-to-go IDS/IPS (Intrusion Detection/Prevention System) linux
distribution based on the multi threaded Suricata [129] IDS/IPS engine and Snorby [130],
the top notch web application for network security monitoring. Smooth-Sec is built on
Ubuntu 10.04 LTS using the TurnKey Core base as development platform. Functionality
is the key point that allows deploying a complete IDS/IPS System up and running out of
the box within a few minutes, even for security beginners with minimal Linux experience.

2.8 Observations on the State of the Art

Intrusion detection has never been easy nor will be. Even current state-of-art IDS is still
far from being as accurate as other security protections. For any security protection, it’s
impossible to achieve 100 percent security. For IDS, the fact is very true. Even though IDS has drastically improved over time, many IDS still suffer from the difficulty of detecting suspicious activities in severe network environments and from the inaccuracy of detection mechanisms. As a result, many IDSs often generate intolerable quantity of false negative and false positive. Numerous characteristics of modern computer network make it difficult to design and implement novel IDS.

- Most research concentrates on the construction of operational IDSs, rather than on the discovery of new and fundamental insights into the nature of attacks and false positives.
- Much research is based on strong assumptions that complicate practical application.

Up to now, data mining in intrusion detection focuses on a small subset of the spectrum of possible applications.

2.9 Challenges of Intrusion Detection

The intrusion detection problem has three basic competing requirements: speed, accuracy, and adaptability. The speed problem represents a quality of service issue. The more analysis (accurate) the detector, the higher is the computational overhead. Conversely, accuracy requires sufficient time and information to provide a useful detector. Moreover, the rapid introduction of both new exploits and the corresponding rate of propagation require that detectors be based on a very flexible/scalable architecture. The most relevant challenges in the area of intrusion detection that motivates the research work in this area can be summarized as:

1. **False alarms**: One of the most important challenges in intrusion detection is detection accuracy [34, 131]. The high rate of false positives and false negatives intrusion detection tools generate does not help system administrators simplify their daily tasks. On the contrary, they may eventually opt for disabling these tools rather than having to interpret the alarms reported by the security system that may, in fact, not be reflecting the truth.

2. **Performance**: Data filtering and screening is a computationally demanding task that is central to intrusion detection [132]. Providing a system with real-time
monitoring capabilities implies good search and pattern matching algorithms as well as very fast processing of network traffic, audit trails, etc. that guarantees no data element goes through without being scrutinized. Given the detailed checking needed for intrusion detection, a lot of resources are consumed and the performance of the hosting system ends up being severely affected. As a consequence, the detection task is moved to be run off-line where its benefits are considerably reduced [34].

3. **Amount of data:** The amount of data available within a single host may render the intrusion detection problem intractable. For network-based intrusion detection systems, the amount of traffic going through a gateway, for instance, can be immensely large making it practically impossible for a security scanner to check every single packet [34]. Similarly, audit trail facilities can generate huge log files which also limit the possibility of real-time detection. Intrusion detection is a needle-in-the-haystack problem as the amount of normal activity information may be huge compared to the pieces of irregular activity that need to be found.

4. **Switched networks:** Modern networks utilize more intelligent devices to route packets. Many intrusion detection prototypes are based on the ability to configure a network device in promiscuous mode in order to observe a larger number of packets. With modern network technology, this monitoring approach may no longer be as successful. A network switch only forwards a packet to its intended recipient to optimize bandwidth use. Network-based intrusion detection systems need to be strategically placed now in order to protect as many hosts as possible inside a LAN. The filtering functions of a switch which represent a performance advantage are in fact a challenge to the security of networked systems [17].

5. **Encryption:** Encrypted data cannot be easily interpreted unless the decryption key is available. The evolution of the Internet has witnessed the development of cryptographic protocols and tools that protect the privacy of data. Encrypted file transfers, encrypted interactive sessions, and encrypted email all eliminate the possibility of meticulous security analysis [17]. As more companies become security-conscious the use of encryption is increasing and will certainly increase
even more [30, 32]. Intrusion detection tools are currently unable to deal with this problem as they are blind to encrypted information.

6. **Security of intrusion detection technologies:** By definition, a security tool must be secure. Otherwise it cannot be trusted and its results are worthless. As software products, intrusion detection systems face the same security challenges other applications do (e.g., poor implementation and a weak design). In fact, not much attention has been put into the secure design of this type of software [133]. Only a few systems can be considered moderately secure but the vast majority is prone to many types of security attacks.

7. **Reaction to incidents:** Although it is not part of the traditional definition of intrusion detection, response to incidents is a very desirable feature that a few intrusion detection systems have actually tried to implement. Taking action based on incorrect information can have serious consequences and the problem of automated response has remained unexplored due to the low levels of detection accuracy intrusion detection tools currently display [133].

Above discussions review the related work. Current approaches for intrusion detection have the following problems.

- Current approaches often suffer from relatively high false-alarm rates, whereas they have high detection rates. As most network behaviours are normal, resources are wasted on checking a large number of alarms that turn out to be false.
- Their computational complexities are oppressively high. This limits the practical applications of these approaches.
- The intrusion detection systems works on attack signatures. Attack signatures are attack patterns of previous attacks. The signature database needs to be updated whenever a different kind of attack is detected and the fix for the same is available.

**Summary**

There are various approaches to build intrusion detection system. Some of them are statistical based approaches and data mining approaches. Pattern matching algorithms
solve the general pattern matching problem for intrusion detection. Various machine learning approaches have been proposed in an attempt to improve on the generic signature-based IDS. Generic machine learning approaches include clustering or data mining in which case the data is effectively unlabeled. Specific examples of such algorithms include artificial immune systems as well as various neural network and clustering algorithms. Number of machine learning approaches, including: decision trees, neural networks and genetic programming have been proposed. Learning algorithms can be categorized as supervised, unsupervised and semi-supervised. All the three approaches can be used for intrusion detection. Supervised machine learning methods require labelled data for training. Semi-supervised learning methods can leverage unlabeled data in addition to labelled ones. Semi-supervised learning is learning from a combination of both labelled and unlabeled data. Unsupervised algorithms learn from unlabeled examples. Attack taxonomies and the resulting classifications are of interest which envisages using a classification of attacks to identify the input to the evaluation of IDSs. Computer security attacks and vulnerabilities have been classified in many ways such as DOS, R2L, U2R, and Probing. The intrusion detection problem has three basic competing requirements: speed, accuracy, and adaptability.

An Intrusion Detection System is a crucial part of the defensive operations that complements the static defences such as firewalls. Essentially, intrusion detection systems search for signs of an attack and flag when an intrusion is detected. According to the detection methodology, intrusion detection systems are typically categorized as misuse detection and anomaly detection systems. From a deployment perspective, they are be classified as network based or host based.