Chapter 2
Detection Approaches

The basic principle of intrusion detection is based on the assumption that intrusive activities are noticeably different from normal ones and thus are detectable [16]. Many intrusion detection approaches have been suggested in the literature since Anderson's seminal report [5]. Traditionally these approaches are classified into three categories: misuse detection, anomaly detection and specification-based detection. Anomaly based intrusion detection approaches are dedicated to establishing a model of the data flow that is monitored under normal conditions without the presence of any intrusive procedures. In contrast, misuse detection approaches aim to encode knowledge about patterns in the data flow that are known to correspond to intrusive procedures in form of specific signatures. In specification based detection approaches, security experts predefine the allowed system behaviors and thus events that do not match the specifications are labeled as attacks. In this chapter we discuss these different approaches in detail and summarize some representative examples in each category.

2.1 Misuse Detection

The study of misuse detection began with Anderson's report in 1980. Intrusions are detected by matching actual behavior recorded in audit trails with known suspicious patterns. While misuse detection is fully effective in uncovering known attacks, it is useless when faced with unknown or novel forms of attacks for which the signatures are not yet available. Moreover, for known attacks, defining a signature that encompasses all possible variations of the attack is difficult. Any mistakes in the definition of these signatures will increase the false alarm rate and decrease the effectiveness of the detection technique.

Figure 2.1 illustrates a typical misuse detection model. The model consists of four components: namely, data collection , system profile , misuse detection and response. Data are collected from one or many data sources including audit trails, network traffic, system call trace, etc. Collected data are transferred into a format that is understandable by the other components of the system. The system profile is used to characterize normal and abnormal behaviors. The profiles characterize what a normal subject behavior should be and what operations the subjects typically would perform or do not perform on the objects. The profiles are matched with actual system activities and reported as intrusions in case of deviations.

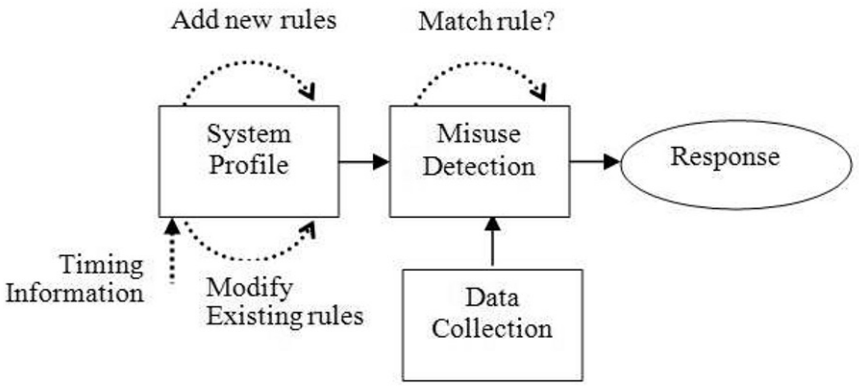


Fig. 2.1 A typical misuse detection system

Four classes of techniques are commonly used to implement misuse detection, namely pattern matching, rule-based techniques, state-based techniques, and data mining. We discuss in detail these techniques and sample systems in the rest of this section.

2.1.1 Pattern Matching

Pattern matching based intrusion detection approaches are commonly used in the network based intrusion detection systems in which attack patterns are modeled, matched and identified based on the packet head, packet content or both. Attack patterns could also be established in host-based intrusion detection systems through concatenating the words representing the system calls in a system audit trail. With the continual emerging of new types and varied forms of attacks the number of signatures is constantly growing, thus making the pattern matching more expensive in terms of the computational cost. In order to address this limitation, Abbes et al. propose a method combining a novel protocol analysis approach with traditional pattern matching to improve the performance of pattern matching when looking for attack signatures [1]. The protocol analysis checks patterns in specific parts of the packet rather than in the entire payload and it is implemented based on the construction of a decision tree. The biggest advantage of this approach is that it can reduce the search space for patterns that results in a fast search. Moreover, the approach has the potential to reduce the number of false positives since patterns are matched only in the extracted protocol fields. The performance evaluation of pattern matching based intrusion detection approaches is studied by Kreibich and Crowcroft [6], in which a workload model is proposed to provide reasonably accurate estimates compared to real workloads. The model attempts to emulate a traffic mix of different applications, reflecting the characteristics of each application and the way these applications interact with the system. The model has been implemented as part of a traffic generator that can be extended and tuned to reflect the needs of different scenarios.

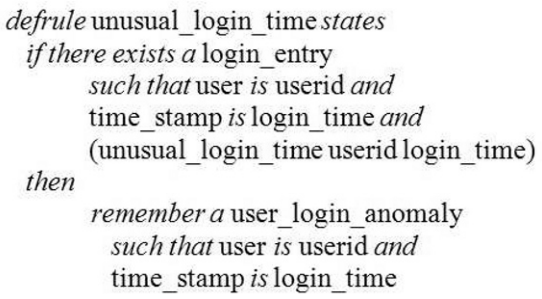
2.1.2 Rule-based Techniques

Rule-based expert system is one of the earliest techniques used for misuse detection. Expert systems encode intrusive scenarios as a set of rules, which are matched against audit or network traffic data. Any deviation in the rule matching process is reported as an intrusion. Examples of rule-based systems include MIDAS (Multics Intrusion Detection and Alerting System) [65], IDES (Intrusion Detection Expert System) [53], and NIDES (Next-generation Intrusion Detection Expert System) [3, 4].

2.1.2.1 MIDAS

MIDAS was designed and developed by the National Computer Security Center (NCSC) to monitor intrusions for NCSC's networked mainframe, Dockmaster. It uses and analyzes audit log data by combining the expert system technology with statistical analysis. MIDAS uses the Production Based Expert System Toolset (P-BEST) [51] in discriminating and implementing the rule base, which is written in LISP language.

The structure of the rules in the P-BEST rule base includes two layers. The first (lower) layer is used to match certain types of events such as number of user logins, and then fires new events by setting up a particular threshold of suspicion. Rules in the second (higher) layer process these suspicions and decide whether the system should raise an alert or not. Figure 2.2 illustrates an example of MIDAS rule. The rule defines an intrusion scenario involving some unusual login time. It determines whether the time when the user logins is outside normal hours or not. The rule also illustrates that an unusual behavior does not necessarily stands for an intrusion.



```
defrule unusual_login_time states
  if there exists a login_entry
       such that user is userid and
       time_stamp is login_time and
       (unusual_login_time userid login_time)
  then
       remember a user_login_anomaly
         such that user is userid and
         time_stamp is login_time
```

Fig. 2.2 Unusual login time rule

2.1.2.2 IDES

IDES is one of the early implementation of the basic ideas and notions underlying intrusion detection. The IDES model results from Denning's seminal paper [16], which provides a mathematical codification of intrusion detection mechanisms. The model is based on the assumption that normal interactions between subjects (e.g. users) and objects (e.g. files, programs, or devices) can be characterized, and also that users always behave in a consistent manner when they perform operations on the computer system. These usages can be characterized by computing various statistics and correlated with established profiles of normal behaviors. New audit records are verified by matching known profiles for both subjects and their corresponding groups. Deviations are then flagged as intrusions. To improve the detection rate, IDES monitors the subject depending on whether the activity happens on an *on* or *off* day since user activities on different day types are usually different. For example, activities for normal users on a working day may be abnormal on an off-working day.

IDES uses P-BEST to describe its rule base consisting of two types of rules: generic rules and specific rules. Generic rules can be used for different target systems and specific rules are strictly dependent on the operating system and the corresponding implementation. IDES architecture consists of three main components, namely audit database, profiles database and the system security officer (SSO) user interface.

2.1.2.3 NIDES

NIDES, the successor of IDES, is the acronym for the Next-generation Intrusion Detection Expert System. NIDES is a hybrid intrusion detection system consisting of a signature-based expert system component as well as a detection component based on statistical approaches. The expert system improves the old IDES version by encoding more known intrusion scenarios and updating the P-BEST version used. The detection component based on statistical approaches is based on anomaly detection. In these approaches over 30 criteria are used to establish normal user profiles including CPU or I/O usage, command used, local network activity, system errors, etc.

The NIDES system is highly modularized with well-defined interfaces between components. Compared with the IDES, NIDES has higher detection rate since it includes two complementary detection components: intrusions missed by one component maybe caught by the other one.

2.1.2.4 Limitations of Rule-based Techniques

Using rule-based techniques for misuse detection, quite often the burden of extending the rule-base as new intrusion scenarios are discovered falls on the shoulder of the security officer. Moreover, developing intrusion scenarios is not an easy task and requires a certain level of expertise and security insight and awareness. In addition, determining the relations between rules is difficult. When many related rules are included in an expert system, correctness of rules is difficult to verify due to the interactions among these rules. Consequently, in practical settings, most of the rule bases are outdated and quickly become obsolete.

2.1.3 State-based Techniques

State-based techniques detect known intrusions by using expressions of the system state and state transitions. State models simplify the specification of patterns for known attacks and can be used to describe attack scenarios easier than rule-based languages such as P-BEST. In state-based techniques, activities contributing to intrusion scenarios are defined as transitions between system states, and thus intrusion scenarios are defined in the form of state transition diagrams. Figure 2.3 depicts a generic state diagram; a node represents a system state, and an arc stands for an action.

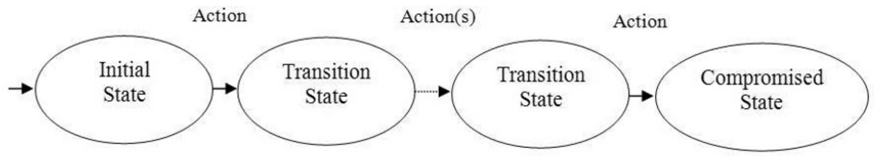


Fig. 2.3 Generic state transition diagram

The state of the system is a function of users or processes. Intrusion scenarios defined by the state transition diagram include three types of states, namely initial state, transition state and compromised state. An initial state refers to the beginning of the attack, while a compromised state stands for the successful completion of the attack. Transition states correspond to the successive states occurring between an initial state and a compromised state. An intrusion occurs if and only if a compromised state is finally reached. In the following section, we present two examples of state-based techniques, namely the state transition analysis tool proposed by Ilgun and colleagues [31, 32], and the IDIOT system based on colored petri-nets proposed by Kumar et al. [43, 41, 42, 40].

2.1.3.1 UNIX State Transition Analysis Tool (USTAT)

The UNIX State Transition Analysis Tool (USTAT) is based on the assumption that all attackers start from an initial state where they possess limited authorization to access a target system, and then after completing some operations on the target system, they acquire some previously unauthorized capabilities. USTAT is a mature prototype implementation of the state transition analysis technique for intrusion detection. It monitors the system state transition from safe to unsafe by representing all known vulnerabilities or intrusion scenarios in the form of a state transition diagram.

In USTAT, over 200 audit events are represented by ten USTAT actions, such as read(file_var), modify_owner(file_var), where the parameter file_var stands for the name of certain files. Known attacks are modeled as a sequence of state transitions that lead from an initial limited authorization state to a final compromised state. An inference engine in USTAT maintains a state transition table and determines whether the current action will cause a state transition to its successor state by matching the next state with the state transition table. Once the next new state is matched to the final state of the transition table, the intrusion alarm is raised.

USTAT is a host-based intrusion detection based on the State Transition Analysis Technique. Some other members of STAT family described in [84] include: NetSTAT is designed for real-time state-transition analysis of network data. WebSTAT and logSTAT are two other STAT family members that operate at the application level. Both of them apply STAT analysis to the events information stored in log files. WebSTAT parses the logs produced by Apache web servers, and logSTAT uses UNIX syslog files as input. Other examples for the STAT approach include [83, 79, 80, 82, 81].

Two additional assumptions underly the state transition analysis technique. First, intrusive behavior must have a visible effect on the system state, and second, this visible effect must be recognizable without using any external knowledge outside the system itself. Not all possible intrusions, however, satisfy these assumptions. For instance, passive listening of broadcast network traffic violates these assumptions. As a result, this technique carries the potential of missing a lot of intrusions that are either not recorded by the audit trail or not represented by state transition diagrams.

2.1.3.2 Colored Petri-nets

IDIOT, the acronym of Intrusion Detection In Our Time, is a state-based misuse detection system which uses pattern matching techniques based on the colored petri-nets (CPN) model. Intrusion scenarios are encoded into patterns in IDIOT, and incoming events are verified by matching them against these patterns. In the CPN model used in the implementation of IDIOT, a guard represents an intrusive signature context and the vertices represent system states. The selected CPN model is referred to as colored petri automata (CPA). The CPA defines a strict declarative specification of intrusions and specifies what patterns need to be matched instead of how to match them.

Figure 2.4 illustrates a simple example of CPA describing the following intrusion scenario: if the number of unsuccessful login attempts exceeds four within one minute report an intrusion. The combination of arrows and vertical bar stands for a transition between system states. For example, the transition from states S1 to S2 occurs when there is a token in S1; this stands for an unsuccessful login attempt. The time of first unsuccessful login attempt is saved in the token variable T1. The transition from S4 to S5 happens if there is a token in S4. The time difference between this and the first unsuccessful login attempt should be more than one minute, otherwise the system state is transferred to the final state S5, in which an alarm will be generated.

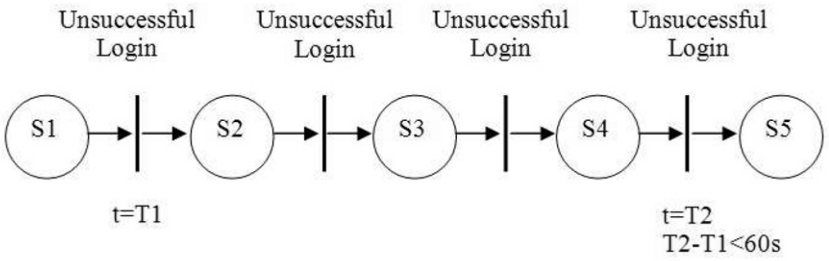


Fig. 2.4 Example of CPA illustrating four failed login attempts within one minute

It is suggested that this technique has several advantages. First, since the intrusion signatures are written in a system independent script, they can be exchanged across different operating systems and different audit logs. Second, the IDIOT system achieves an excellent real-time performance; only 5-6% CPU overhead was reported when it scanned for 100 different patterns. Third, multiple event sessions can be processed independently and then corresponding detection results are analyzed together to make the final decision.

The main limitation of this technique is that it can only detect known vulnerabilities. Also, translating known intrusions into patterns is not always easy. In addition, strict declarative expression about the intrusive patterns leads to a potential problem, that is, sophisticated attackers can easily bypass the detection system by changing their attack strategies.

2.1.4 Techniques based on Data Mining

In recent years, data mining techniques have been applied to network and host audit data for building misuse detection models [48, 49, 47]. In this case, intrusion detection is considered as a data analysis process, in which data mining techniques are used to automatically discover and model features of user's normal or intrusive behaviors. It is reported that three types of algorithms are particularly useful for mining audit data, namely classification, link analysis and sequence analysis.

Classification algorithms such as decision tree generate classifiers by learning based on a sufficient amount of normal or abnormal audit data. New audit data are labeled as either normal or abnormal according to the classifier. Link analysis determines the relation between fields in the audit database records and normal profiles are usually derived from these relations. Sequence analysis is used to find sequential patterns in audit data and embed these patterns into intrusion detection models.

Data mining based techniques performed very well in detecting known attacks during the 1998 DARPA intrusion detection competition. However, just like other misuse detection techniques, they operated fairly poorly on detecting new attacks. In addition, applying data mining techniques requires labeling the training data set, which makes the detection process error-prone, costly and time consuming.

2.2 Anomaly Detection

Different from misuse detection, anomaly detection is dedicated to establishing normal activity profiles for the system. It is based on the assumption that all intrusive activities are necessarily anomalous. Anomaly detection studies start by forming an opinion on what the normal attributes for the observed objects are, and then decide what kinds of activities should be flagged as intrusions and how to make such particular decisions.

A typical anomaly detection model is illustrated in Figure 2.5. It consists of four components, namely data collection, normal system profile, anomaly detection and response. Normal user activities or traffic data are obtained and saved by the data collection component. Specific modeling techniques are used to create normal system profiles. The anomaly detection component decides how far the current activities deviate from the normal system profiles and what percentage of these activities should be flagged as abnormal. Finally, the response component reports the intrusion and possibly corresponding timing information.

The primary advantage of anomaly detection is its capability to find novel attacks; as such it addresses the biggest limitation of misuse detection. However,

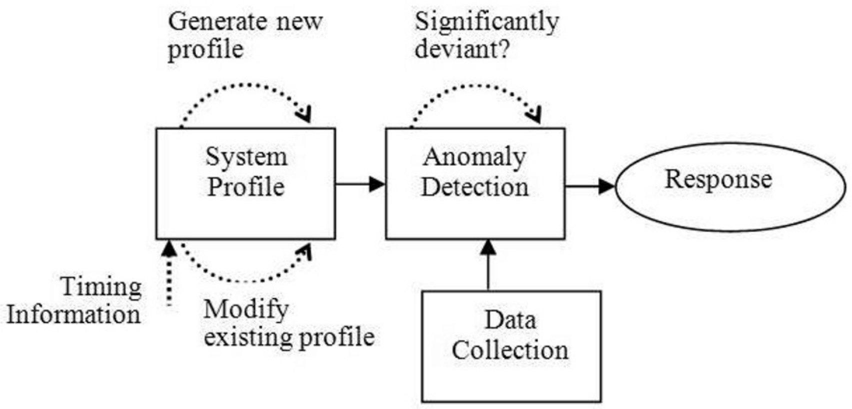


Fig. 2.5 A typical anomaly detection system

due to the assumptions underlying anomaly detection mechanisms, their false alarm rates are in general very high. Specifically, the main reasons for this limitation include the following:

1. The user's normal behavior model is based on data collected over a period of normal operations; intrusive activities missed during this period are likely to be considered as normal behaviors.
2. Anomaly detection techniques can hardly detect stealthy attacks because this kind of attacks are usually hidden in large number of instances of normal behaviors. Moreover, the types of parameters used as inputs of normal models are usually decided by security experts. Any mistake occurring during the process of defining these parameters will increase the false alarm rate and decrease the effectiveness of the anomaly detection system.

As a result, the design of the detection methods and the selection of the system or network features to be monitored are two of the main open issues in anomaly detection.

Many anomaly detection techniques have been proposed in the literature. These range from advanced statistical models to artificial intelligence and biological models based on human immune systems. Although it is difficult to classify these techniques we can divide them into four categories based on previous surveys on anomaly detection systems [7, 46, 35, 56, 27, 55, 72]. These include advanced statistical models, rule-based models, learning models, biological models, and signal processing techniques based models.

2.2.1 Advanced Statistical Models

Denning proposed in a seminal paper the earliest theoretical characterization of anomaly detection. Her detection framework, which is based on basic statistical analysis, consists of eight components: subjects, objects, audit records, statistical metrics, statistical models, profiles and profile templates, anomaly records and activity rules. A subject could be a user, a process, or system itself. The object is the receptor of actions and could be entities such as files, programs and messages. Audit records are m-tuples that represent a set of actions performed by subjects on objects. Once an event generator creates the audit records, the statistical model matches it with the appropriate profile and then makes decisions regarding the profile update, the abnormal behavior checking, and the report of detected anomalies. The activity profile includes various of variables measuring the behavior of the system based on predefined statistical metrics. The basic idea in Denning's model appear with little modification in many intrusion detection systems [40], such as Haystack [70], NIDES [3, 4] and EMERALD [61]. We provide an overview of these systems in the rest of this section.

2.2.1.1 Haystack

The haystack system was designed and implemented for the detection of intrusions in a multi-user Air Force computer system [70]. Statistical techniques are used to detect anomalous activities. A set of features such as the amount of I/O, CPU utilization, number of file accesses are observed and then the normal range of values for these features are defined. Activities falling outside these ranges are reported as intrusions.

Haystack is one of the earliest anomaly detection systems based on statistical models. It uses a very simple statistical model in which each feature is assigned a weight by the SSO. The main weakness of Haystack is that no test is conducted to verify if the weight value is sensitive to intrusion patterns and furthermore no explanation is given as to how the weight value is assigned. Moreover, the underlying assumption that features are statistically independent is usually not satisfied in practice.

2.2.1.2 NIDES

As indicated in the earlier section, NIDES includes a statistical anomaly detector as well. The audit information collected consist of user names, names of files accessed, elapsed user CPU time, total number of files opened, number of pages read from secondary storage, identities of machines onto which user has logged, etc. Statistics are computed from the collected audit information. NIDES stores only statistics related to frequencies, means, variances or covariance of measures instead of the total audit data. Given measures in NIDES, if the point in the m-space of measures is far away from the expected value, the corresponding event represented by the point is considered anomalous.

2.2.1.3 EMERALD

The Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) is an environment for anomaly and misuse detection. It consists of a signature analysis component and a statistical profile-based anomaly detection component. The anomaly detector is based on the statistical approach where the events are labeled as intrusive if they are largely deviant from the expected behavior. In EMERALD more than 30 different criteria including CPU and I/O usage, commands used, local network activity, and system errors are used to build these expected user profiles. An EMERALD monitor is either passive through reading activity logs or network packets or active via probing for gathering a normal event. The analytical results can be exchanged asynchronously between different client monitors operated on different layers (e.g. domain layer, enterprise layer, etc.) [58]. Moreover, each monitor has an instance of the EMERALD resolver, a countermeasure decision engine fusing the alerts from its associated analysis engines and invoking response handlers to counter malicious activities.

More recently a lot of intrusion detection approaches based on statistical models have been proposed. Most of them model system or network behaviors on different input sources such as the application level anomaly detection approaches proposed by Kruegel et al. [39] in which sources of data are application log files, the user level anomaly detection approach presented by Lane [44] in which sources of data are UNIX command data sets collected from real human users under normal working conditions, the network level anomaly detection methods proposed by Bradford [9, 10], Thatton [75] and Siris [69] in which network flow and SNMP data are used for identifying traffic anomalies.

2.2.2 Rule based Techniques

Several anomaly detection models proposed in the literature were implemented using rule-based techniques. We present in this section four examples of such models, namely Wisdom & Sense, NSM, TIM and NADIR.

2.2.2.1 Wisdom & Sense (W&S)

Wisdom & Sensor is a unique approach to anomaly detection, which consists, as the name indicates, of two components: wisdom and sense [78]. The wisdom component consists of a set of rules describing normal behaviors of the system based on historical audit data. About 10,000 records per user are read from the file in order to create the rules. The sense component is an expert system based on the previous rules, which verifies whether the subsequent audit data violates the rule base or not. The system security officer is then alerted when the sense detects some anomalous behavior. In the implementation of W&S, the Figure of Merit (FOM) metric is used to compute an anomalousness score for each thread. The FOMs values for several events are summed and compared against a threshold. When the sum value is above the threshold, corresponding events are flagged as anomalies by the system.

2.2.2.2 Network Security Monitor (NSM)

Network Security Monitor (NSM) is a rule-based intrusion detection system targeting anomalous network activities. It is the first intrusion detection system using network traffic directly as the primary source of data [28]. All network traffic passing through a broadcast LAN are observed by NSM. Since network traffic is based on standard protocols such as TCP/IP, telnet, SMTP, etc., the traffic information on heterogeneous hosts is based on a consistent format.

Network traffic profiles are created by generating connection vectors derived from the network data. Host vectors and connection vectors are the main input to the expert system in NSM. The expert system illustrates the data-path with which systems are expected to communicate and the specifications of higher-level application layer protocols. Intrusions are decided according to analysis results of the expert system. The final result reported to the SSO consists of a connection vector and corresponding suspicion level. The suspicion level measures the likelihood that a particular connection represents some intrusive behaviors.

2.2.2.3 Time-based Inductive Machine (TIM)

Time-based Inductive Machine (TIM) models user's normal behavior patterns by dynamically generating activity rules using inductive generalization [73]. TIM discovers normal behavior patterns in a sequence of events instead of a single event. Each rule represents a model that can predict the next event from a given sequence of events. If user's behavior matches with previous event sequences and the actual next event is not included in the predicted event sets, the user's behavior is considered as intrusive. The major limitation of TIM is that the rules only model adjacent events. When many applications are executed at the same time, events occurring in one application may be interleaved with events occurring in other applications, in which case the rules generated by TIM may not accurately identify user's normal behaviors. Moreover, events that cannot match any previous event sequences in the training data set always fire an intrusion alarm, which most likely may correspond to some false alert.

2.2.2.4 NADIR

NADIR (Network Anomaly Detection and Intrusion Reporter) was developed at Los Alamos National Laboratory to monitor their internal computer networks [33, 29]. It collects audit information from three different kinds of service nodes, namely network security controller, common file system and security assurance machine. Network security controller provides user authentication and access control; common file system stores the data, and security assurance machine records attempts to degrade the security level of the data. The audit information collected for each event include a unique user ID associated with the subject, the date and time of the event, an accounting parameter, the error code, a flag indicating whether the corresponding action is successful or not, and the description of the event.

Individual user profile is computed on a weekly basis from the audit database. The user profiles are compared with a set of expert system rules in order to detect possible deviations in users' behaviors. The expert rules are mainly derived from the security policies and the statistical analysis of previous audit logs. Each rule is assigned a level-of-interest. If the corresponding sum of level-of-interest for a user is too big the intrusion alarm is raised.

2.2.3 Biological Models

Several anomaly detection models inspired from biological principles have been proposed in the literature. One of the earliest works in this area was authored by Forrest et al. They studied the analogy between human immune system's capability of distinguishing *self* from *nonself* and intrusion detection system [24, 22, 23, 18, 85]. In the human systems, T cells are created in the thymus and then a censoring process happens. If T cells bind with proteins or peptides (sub-units of proteins), then they will be transferred because they bind with themselves (*self*). Some T cells that do not bind with proteins are released to monitor the foreign material (*nonself*) of the body. A hypothesis is that those T cells that bind with foreign materials will then be removed from human systems.

When applying the *self-nonself* technique to intrusion detection, behaviors of the system are viewed as a string. This string is divided into sub-strings of equal length k. *Self* contains some of the possible 2^k strings. The rest of the sub-strings $2^k - self$ or the complement of *self* is called *nonself*. *Nonself* contains some detector strings, which are of length and does not exist in the collection of *self*. In the practical implementation, current behaviors of the system are first divided into sub-strings of length k and then are periodically compared with detector strings in *nonself*. A match indicates a possible intrusion or an unexpected change.

Nonself can be generated through various ways. A typical implementation for this is to randomly generate strings of length k, and then remove any string that is part of the collection of *self*. Consequently, a sub-string that is not in *self* is used as a detector and is added to *nonself*. The size of *nonself* determines the effectiveness of detecting anomalous behaviors; the larger the size of *nonself* the more likely intrusion will be detected.

Forrest et al. point out that perfect matching between strings is rare. Thus, they define a partial matching rule: Given any two strings p and q, expression $match(p,q)$ is true if p and q match in at least r contiguous locations. For example, considering two strings x and y, where $x = CEFG\underline{OPRT}SY$ and $y = BABS\underline{OPRT}TZ$, with $r > 4$, $match(x,y)$ is false, but with $r < 5$, $match(x,y)$ is true. Since perfect matching is rare, the partial matching rule increases the likelihood of detecting anomalous behaviors.

Another interesting biological method for intrusion detection proposed by Yu et al. is based on the DNA sequence [89]. Yu et al. define DNA sequences for a computer system based on the knowledge that the DNA characterizes the make up of human body and that any anomaly in tissues can be reflected in a particular DNA sequence. Any change in the behavior patterns of the computer system may be traced to the change of DNA sequences that can be either normal or abnormal. A standard back-propagation neural network was used to train normal DNA sequences for network traffic, and then a UDP Flood attack was successfully detected based on the DNA sequence for normal network traffic. Although encouraging, this preliminary result needs to be extended in order to define a more complete DNA scheme for computing systems.

More recently, Ahmed and Traore proposed the use of behavioral biometric for detecting masqueraders [2]. Biometrics has so far been used only for authentication not for intrusion detection. The framework proposed by Ahmed and Traore uses mouse and keystroke dynamics both of which are behavioral biometrics that can be collected passively and checked dynamically. The results reported in [2] seem encouraging.

2.2.4 Learning Models

Learning models incorporate learning capabilities in intrusion detection process, using artificial leaning techniques. In recent years, learning techniques have been widely used in anomaly detection since the self-learning techniques can automatically form an opinion of what the subject's normal behavior is. According to whether they are based on supervised or unsupervised learning techniques we divide the anomaly detection schemes into two categories: unsupervised and supervised. Supervised anomaly detection establishes the normal profiles of systems or networks through training based on labeled data sets. In contrast, unsupervised anomaly detection attempts to detect intrusions without using any prior knowledge of attacks or normal instances. The main drawback of supervised anomaly detection is the need of labeling the training data, which makes the process error-prone, costly and time consuming, and difficult to find new attacks. Unsupervised anomaly detection addresses these issues by allowing training based on unlabeled data sets and thus facilitating online learning and improving detection accuracy. Unsupervised anomaly detection is relatively new compared with supervised anomaly detection schemes. We discuss both approaches in the rest of this section.

2.2.4.1 Supervised Anomaly Detection

In order to illustrate supervised anomaly detection techniques we summarize in the following section two of the proposed approaches, namely neural network and evolutionary approaches based on genetic algorithms.

Using Neural Network

Hyperview is an early attempt for intrusion detection using neural network, which consists of two components [15]. The first component is an expert system, which monitors audit trails for known intrusion signatures. The second component uses neural network to learn user behaviors; it then fires an alarm when there is a deviation between current behaviors and learnt behaviors. The use of neural network in Hyperview assumes that the audit data consist of multivariate time series, since user behavior exhibits a dynamic process executing an ordered series of events. In the implementation of Hyperview, a recurrent network is used for training and the time series are mapped with the neural network. The output is then selected as the next input to the system until the training process is terminated.

Another example of intrusion detection system using neural network is proposed by Ghosh [25]. A back-propagation network is used for training. In the practical implementation, Ghosh uses the program internal state and the input program as the input of back-propagation networks. The experimental results show that this approach increases the performance of anomaly detection by randomly generating data as anomalous input. However, choosing input parameters is not easy. Any mistake in input data will increase the false alarm rate. In addition, how to initialize the weights remains unclear.

The recent work on applying neural networks for detecting network anomalies is illustrated by Lei and Ghorbani in [50], in which a novel approach for detecting network intrusions is presented based on a competitive learning neural network. The self-organizing map (SOM) has been applied in anomaly detection for several years and it implicitly prepares itself to detect any aberrant network activity by learning to characterize the normal behaviors. However, the SOM has a significant shortage that is the number of neurons affects the network's performance. Increasing the number of output nodes will increase the resolution of the map, but the computation time will dramatically increase. As a result, Lei and Ghorbani proposed an efficient clustering algorithm based on the competitive neural networks. Experimental evaluations showed that the proposed approach obtained a similarly accurate detection rate as the SOM does, while using only one forth of the computation time of the SOM.

Using Genetic Algorithm

The genetic algorithm is an iterative search technique based on the theory of Darwinian evolution applied to mathematical models. It operates on a population of individuals in which each individual is a potential solution to a given problem. After the initial population is randomly generated the algorithm circularly evolves the population through three basic operators: selection operator, crossover operator and mutation operator until the best individual is obtained.

Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis (GASSATA) is one of the earliest attempts for using genetic algorithm (GA) for intrusion detection [57]. In GASSATA, Me defines an n-dimensional hypothesis vector H, where $H_i = 1$ if attack i is taking place according to the hypothesis, otherwise $H_i = 0$. As a result, the intrusion detection becomes the problem of finding the H vector that maximizes the product $W \times H$, subject to the constraint $(AE \times H)_i \leq O_i$; where W is a n-dimensional weight vector, AE is an attacks-events matrix, and O is the observed n-dimensional audit trail vector. Each individual in the population corresponds to a particular H vector. The fitness function is defined as follows,

$$Fitness = \sum_{i=1}^{n} W_i \times I_i \tag{2.1}$$

Where I_i stands for an individual. The experimental evaluation showed that GA for intrusion detection had a low false alarm rate and the likelihood of detecting misuse behaviors was 0.996. However, the major limitation of GASSATA is that it cannot locate attacks precisely.

Another attempt using GA for intrusion detection is made by Chittur [13]. Chittur uses a certainty formula C_i to classify whether a record is an intrusive behavior or a normal behavior. C_i is defined as follows,

$$C_i(x) = \sum_{j=1}^{n} \Re_{ij} \times x_j \tag{2.2}$$

Where $\Re_i j$ is the Ephemeral Random Constant-based coefficient for attribute x_j, and n is the number of attributes. A threshold value for C_i is established and any certainty value exceeding this threshold value is classified as a malicious attack.

The following fitness function is used:

$$F(\delta_i) = \frac{\alpha}{A} - \frac{\beta}{B} \tag{2.3}$$

Where δ_i refers to the individual; α means the number of correctly detected attacks; and, A stands for the number of total attacks; β refers to the number of false positives and B is the total number of normal connections. The range of the fitness value is from -1 to 1. A high detection rate $\frac{\alpha}{A}$ and a low false positive rate $\frac{\beta}{B}$ will yield a high fitness value for an individual.

The experimental results show that GA successfully generates an accurate empirical behavior model from the training data. However, the major limitation of this approach is that an improper threshold value might easily lead to a high false alarm rate in detecting new attacks.

More efforts using GA for intrusion detection are made in [26, 8, 12]. Gomez et al. propose a linear representation scheme for evolving fuzzy rules using the concept of complete binary tree structures. GA is used to generate genetic operators for producing useful and minimal structure modifications to the fuzzy expression tree represented by chromosomes. However, the training process in this approach is computationally very expensive and time consuming. Bridges and Vaughn employ GA to tune the fuzzy membership functions and select an appropriate set of features in their intelligent intrusion detection system. Balajinath and Raghavan use GA to learn individual user behaviors. Active user behaviors are predicted based on past observed user behaviors which can be used for intrusion detection. The training process for both approaches is, however, time consuming.

Crosbie and Spafford propose a framework combining genetic programming (GP) and agent technology for detecting anomalous behaviors in a system [14]. Autonomous agents are used to detect intrusions using log data of network connections. Each autonomous agent is used to monitor a particular network parameter. Autonomous agents whose predication is correct are assigned a higher weight value to decide whether a session is intrusive or not. There are a number of advantages to having many small agents instead of a single large one. However, communications among so many agents lower the detection efficiency. Moreover, an improper primitive for each agent will make the training time long.

2.2.4.2 Unsupervised Anomaly Detection

Several unsupervised learning techniques for anomaly detection have been proposed in recent years. Clustering and outlier detection based techniques are among the most popular approaches suggested so far.

Lankewicz et al. author one of the earliest works in the literature [45]. Specifically, they propose a model applying pattern recognition to the characterization of individual users. The profiles computed with this model can be used for anomaly detection, particularly at the host level. A key aspect of their approach consists of using a variant of k-nearest-neighbor clustering algorithm for data compression and cluster discovery. According to them, this would facilitate real-time detection. Unfortunately, even though the main goals sought by the authors consist of improving detection efficiency and achieving real-time detection, it is not clear whether their model can achieve these goals, since no evaluation data are actually provided.

Staniford et al. propose a *portscan* detector, which is particularly effective against stealthy scans [71]. They propose an architecture that combines an anomaly sensor with a correlator. Firstly, the anomaly sensor computes an anomaly score for packets. Secondly, based on the anomaly scores the correlator uses simulated annealing to cluster packets into activities that are similar. This architecture is similar to ours, particularly since our IP weight metrics play the same role as their anomaly score. Our model, however, is broader and targets a wider range of network attacks, while their model focuses only on *portscan*. As indicated by the authors, formal experiments to measure the detection performance of their model have yet to be completed.

Another early attempt to unsupervised anomaly detection is presented by Eskin [20]. A mixture probability distribution model D is proposed to identify anomalous behaviors. Eskin assumes that each element in the model falls into one of two cases: elements are either anomalous with small probability (λ) or are normal with majority probability $(1-\lambda)$. D is thus composed by normal distribution M and alternate distribution A according to the probability λ, giving $D=(1-\lambda)\times M+\lambda\times A$. EM algorithm is used to estimate D. This method shows a strong ability to detect intrusive system calls when the amount of anomalous system calls consists of less than 5% of the total data. However, it has high false alarm rate when the number of anomalies is much larger than the number of normal instances in the data set.

Portnoy et al. extended the unsupervised anomaly detection approach suggested by Eskin [62]. They use a simple distance-based metrics and a simple variant of single-linkage to cluster the unlabeled data. In order to identify intrusions, they make the following assumptions:

1. The number of normal data instances largely outnumbers the number of intrusions.
2. The data instances include two clusters: intrusive cluster and normal cluster.

The direct consequence of these two assumptions is that normal instances are expected to form larger clusters compared to intrusive ones. And thus, large clusters will be flagged as normal, while small clusters will be considered intrusive. After testing their methodology using 1999 KDDCUP intrusion detection data set, they report an average detection rate of 40-55% with a 1.3-2.3% false positive rate.

Three clustering algorithms are later suggested to improve the accuracy of clustering-based detection in [21]. The first algorithm calculates the density of points near the data point being analyzed. If this point does not belong to a dense region, it will be considered an anomaly. The second algorithm calculates the sum of all the distances to the k-nearest neighbors. If the sum is greater than a threshold, the data point is flagged as an anomaly. The third algorithm is based on the support vector machine paradigm [64]. It solves convex optimization problems to classify the lower regions of support in a probabilistic distribution. Two different data sets are used to evaluate these algorithms and the evaluation result shows that the clustering algorithms have strong ability to detect anomalies. Although the evaluation results are promising, more experiments need to be done with different data sets.

Another well known clustering algorithm for network-based anomaly detection is Y-means [87]. Y-means is a clustering heuristic for intrusion detection which is based on the K-means algorithm and other related clustering algorithms, and overcomes two shortcomings of K-means, namely number of clusters dependency and degeneracy.

Similar with Y-means, a new unsupervised anomaly detection framework for network intrusions is proposed by Lu and Traore in [52], in which a new clustering algorithm named I-means and new anomalousness metrics named IP Weights are included. I-means is an evolutionary extension of k-means algorithm that is composed by a revised k-means algorithm and an evolutionary approach to mixture resolving, which estimates automatically the number of clusters for a set of data. IP Weights allow the automatic conversion of regular packet features into a 3-dimensional numerical feature space, in which the clustering takes place. Intrusion decisions are made based on the clustering result. The offline evaluation with a subset of 1998 DARPA intrusion detection data set shows that the system prototype detects 18 types of attacks out of a total of 19 network attack types; and an online evaluation in a live networking environment shows the system obtains a strong runtime efficiency, with response times falling within a few seconds ranges.

Yamanishi et al. propose an online unsupervised outlier detection approach for network intrusion detection [88]. Their model uses a finite probabilistic mixture model to represent the network data. An online discounting learning algorithm is used to learn the probabilistic model. Based on the learned model, a score is generated for the data, with a high score indicating an outlier. Intrusion detection in this case consists of identifying outliers and reporting them as intrusions. Evaluation of the model is performed using a subset of 1999 KDDCUP intrusion detection data set and reveals detection ratios ranging between 55% to 82%.

There are several other works that attempt to achieve unsupervised anomaly detection using techniques other than clustering or outlier detection. Examples of those include works by Sekar et al [66] and Shyu et al. [68]. Sekar et al. define state machine specifications of network protocols, augmented with statistical information that needed to be maintained to detect anomaly. They claim that the learning component in their method is robust enough to operate without human supervision and thus can be considered as unsupervised anomaly detection. However, they do not provide any experimental evaluation of these claims in their paper. Shyu et al. propose a novel anomaly detection scheme based on principle component classifier. They evaluate their algorithm using 1999 KDDCUP intrusion detection data. The experimental result shows that their approach has better performance than other approaches, such as LOF approach [11] and Canberra metrics [19]. Based on this robust result, they claim that their method would also work with unlabeled training data. However, they have not conducted any experimental evaluation to support this assertion.

2.3 Specification-based Detection

Specification-based detection approaches are neither misuse based, nor anomaly based since they use system behavioral specifications to detect attacks and premise that every well-behaved system execution shall conform to the specification of its intended behavior [77]. Instead of learning system behaviors, in specification-based systems the experts' knowledge determines the operating limits (threshold) of a system. Once the correct (or allowed) system behavior is specified. The events deviating from the specification would generate an alert [37].

Specification-based detection approaches focus on building specified behaviors based on the least privileged principle (or default deny principle). An execution sequence performed by the subject that violates the specification of programs will be considered as an attack. This approach, in theory, can detect unseen attacks that may be exploited in the future [38]. However, specifying the behavior of a large number of privileged programs running in real operating environments is a daunting and difficult task. Even though inductive logic programming is used to automatically synthesize specifications of programs in [36], rigorous validation of so many programs' specifications is still an open issue. Moreover, the formal methods community has been struggling with the same issue for decades without significant success. Hence, in spite of its appealing principles, specification-based detection still remains in infancy.

2.4 Hybrid Detection

Early research works on intrusion detection systems suggested that the intrusion detection capabilities can be improved through a hybrid approach consisting of both signature (misuse) detection as well as anomaly detection [54, 34, 61]. In such a hybrid system, the signature detection technique detects known attacks and the anomaly detection technique detects novel or unknown attacks. Typical recent research works for such a hybrid intrusion detection system are discussed in [76, 90, 60, 17, 30].

In [76], Tombini et al. applied an anomaly detection approach to create a list of suspicious items. Then a signature detection approach is used to classify these suspicious items into three categories, namely false alarms, attacks, and unknown attacks. The approach is based on an assumption that a high detection rate can be achieved by the anomaly detection component because missed intrusions in the first step will not be found by the follow-up signature detection component. Moreover, to make the system working well, it assumes the signature detection component has the potential to identify false alarms. Although the proposed hybrid system missed certain types of attacks, it do reduced the false alarm rate and increased the likelihood of examining most of the alerts.

Zhang et al. [90] proposed a hybrid IDS combining both misuse detection and anomaly detection components, in which a random forests algorithm was applied firstly in the misuse detection module to detect known intrusions. The outlier detection provided by the random forests algorithm is then utilized to detect unknown intrusions. Evaluations with the part of 1999 KDDCUP data set showed that their misuse detection module generated a high detection rate with a low false positive rate and at the same time the anomaly detection component has the potential to find novel intrusions. The whole hybrid system achieved an overall 94.7% detection rate with 2% false positive rate.

Peng et al. propose in [60] a two-stage hybrid intrusion detection and visualization system that leverages the advantages of signature-based and anomaly detection methods. It was claimed that their hybrid system could identify both known and novel attacks on system calls. However, evaluation results for their system were missed in the paper. The work is more like an introduction on how to apply multiple stage intrusion detection mechanism for improving the detection capability of IDS.

Similar with [90], Depren et al. [17] proposed a novel hybrid IDS system consisting of an anomaly detection module, a misuse detection module and a decision support system. The decision support system was used to combine the results of previous two detection modules. In the anomaly detection module, a Self-Organizing Map (SOM) structure was applied to model normal behavior and any deviation from the normal behavior will be classified as an attack. In the misuse detection module, a decision tree algorithm was used to classify various types of attacks. The final system was evaluated with the 1999 KDDCUP intrusion detection data set and experimental results showed that the proposed hybrid approach gave better performance over individual approaches.

Based on an idea of combining the advantages of low false positive rate of signature based IDS and the ability of anomaly detection system (ADS) for detecting new or unknown attacks, Hwang et al. proposed and reported a new experimental hybrid intrusion detection system (HIDS) in [30]. The ADS was built in HIDS, in which anomalies can be detected beyond capabilities of the well known signature based SNORT or Bro systems through mining anomalous traffic episodes from Internet connections. A weighted signature generation scheme was developed to combine ADS with SNORT through modeling signatures from detected anomalies. The HIDS scheme was evaluated with real Internet trace data mixed with 10 days of the 1999 DARPA intrusion detection data set. The obtained results showed HIDS achieves a 60% detection rate, compared with 30% and 22% detection rate acquired by the SNORT and Bro systems separately.

Instead of combining signature detection techniques and anomaly detection techniques, some other hybrid systems fuse multiple anomaly detection systems according to some specific criteria considering the detection capability for each anomaly detection technique is different. The main goal of such a hybrid system is to reduce the large number of false alerts generated by current anomaly detection approaches and at the same time keep an acceptable detection rate. Some examples of such research works are discussed in [86, 74, 59, 67, 63].

In [86], Xiang et al. proposed a multiple-level hybrid classifier utilizing a combination of tree classifiers and clustering algorithms. One of the most interesting ideas of this work is that they fuse both supervised learning (tree classifiers) and unsupervised learning (clustering) techniques. Although supervised learning technique needs a clear label for training, it was claimed in the paper that unsupervised learning might play an essential role on improving the detection rate. With the KDDCUP 1999 data set, they evaluated their approach and compared it with other popular approaches (i.e. MADAM ID and 3-level tree classifiers). Evaluation results showed that their hybrid approach was very efficient in detecting intrusions with an extremely low false negative rate of 3.37%, while keeping an acceptable level of false alarm rate of 9.1%.

Thames et al. proposed in [74] an intelligent hybrid IDS based on Bayesian Learning Networks and Self-Organizing Maps. Testing results with the 1999 KDDCUP data set showed that such a hybrid system could achieve a significant improvement in classification accuracy compared to a non-hybrid Bayesian Learning approach when network-only data was used for classification. Although their detection results were promising, their intelligent system was sensitive to the training data they used because of the inherent limitation of learning systems. The authors discussed this limitation and proposed a possible solution addressing this issue in their future work.

Peddabachigari et al. proposed a hierarchical hybrid intelligent system model (DT-SVM) based on decision trees (DT) and support vector machines (SVM) techniques in [59]. It was assumed that combining the individual base classifiers and other hybrid machine learning paradigms can maximize the detection accuracy and minimize computational complexity. Evaluation results with part of the 1999 KDDCUP data set showed that probing attacks could be detected with a 100% accuracy and moreover the hybrid DT-SVM approach improved and delivered equal performance for all the attack types when compared to a direct DT or SVM approach separately.

Shon and Moon proposed in [67] a new SVM approach (i.e. Enhanced SVM) by combining two existing SVM techniques, namely soft-margin SVM and one-class SVM, in order to provide unsupervised learning capability and to achieve a low false alarm rate. A set of additional techniques had been used to improve the performance of their approach, including creating normal packets profile with Self-Organized Feature Map (SOFM), filtering packets based on Passive TCP/IP Fingerprinting (PTF), selecting features using Genetic Algorithms (GAs) and using the flow of packets based on temporal relationships in data preprocessing. The experimental evaluation with DARPA intrusion detection data sets and a live data set captured from a real network showed that the proposed enhanced SVM approach obtained a low false positive rate similar to that of some real network IDS without requiring pre-labeled data.

Sabhnani and Serpen compared and evaluated nine well known pattern recognition and machine learning algorithm with the 1999 KDDCUP intrusion detection data set in [63]. Based on the performance for each algorithm, they selected three of them for obtaining an optimal detection result, including Multilayer Perceptron (MLP) for probing attacks, K-means for DoS attacks as well as U2R, and Gaussian classifier for R2L attacks. Evaluation results with the union of these three algorithms showed that the hybrid detection system could achieve a better performance than the 1999 KDD Cup's winner.

References

1. T. Abbes, A. Bouhoula, and M. Rusinowitch, *Protocol analysis in intrusion detection using decision tree*, Proceedings of International Conference on Information Technology: Coding and Computing (ITCC), vol. 1, 2004.
2. A.A.E. Ahmed and I. Traore, *Detecting computer intrusions using behavioral biometrics*, Third Annual Conference on Privacy, Security and Trust (PST), 2005.
3. D. Anderson, T. Frivold, and A. Valdes, *Next-generation intrusion detection expert system (NIDES): A summary*, SRI International, Computer Science Laboratory, 1995.
4. D. Anderson, T.F. Lunt, H. Javitz, A. Tamaru, and A. Valdes, *Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES)*, SRI International, Computer Science Laboratory, 1995.
5. J.P. Anderson, *Computer security threat monitoring and surveillance*, (1980).
6. S. Antonatos, K.G. Anagnostakis, and E.P. Markatos, *Generating realistic workloads for network intrusion detection systems*, ACM SIGSOFT Software Engineering Notes **29** (2004), no. 1, 207–215.
7. S. Axelsson, *Intrusion detection systems: A survey and taxonomy*, Tech. Report 99-15, Chalmers University of Technology, Department of Computer Engineering, 2000.
8. B. Balajinath and SV Raghavan, *Intrusion detection through learning behavior model*, Computer Communications **24** (2001), no. 12, 1202–1212.
9. P. Barford and D. Plonka, *Characteristics of network traffic flow anomalies*, Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, ACM New York, NY, USA, 2001, pp. 69–73.
10. Paul Barford, Jeffery Kline, David Plonka, and Amos Ron, *A signal analysis of network traffic anomalies*, Proceedings of the second ACM SIGCOMM Workshop on Internet measurment (Marseille, France), SIGCOMM: ACM Special Interest Group on Data Communication, ACM Press New York, NY, USA, 2002, pp. 71–82.
11. M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander, *LOF: identifying density-based local outliers*, ACM SIGMOD Record **29** (2000), no. 2, 93–104.
12. S.M. Bridges and R.B. Vaughn, *Fuzzy data mining and genetic algorithms applied to intrusion detection*, Proceedings of the Twenty-third National Information Systems Security Conference, National Institute of Standards and Technology, October 2000.
13. A. Chittur, *Model generation for an intrusion detection system using genetic algorithms*, High School Honors Thesis, Ossining High School in cooperation with Columbia University (2001).
14. M. Crosbie and E. H. Spafford, *Applying genetic programming to intrusion detection*, Proceedings of the 1995 AAAI Fall Symposium on Genetic Programming, November 1995.
15. H. Debar, M. Becker, and D. Siboni, *A neural network component for an intrusion detection system*, Proceedings of the 1992 IEEE Symposium on Security and Privacy, 1992, pp. 240–250.
16. DE Denning, *An intrusion-detection model*, IEEE Transactions on software engineering (1987), 222–232.
17. O. Depren, M. Topallar, E. Anarim, and M.K. Ciliz, *An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks*, Expert systems with Applications **29** (2005), no. 4, 713–722.
18. P. D'haeseleer, S. Forrest, and P. Helman, *An immunological approach to change detection: algorithms, analysis, and implications*, IEEE Symposium on Security and Privacy, IEEE COMPUTER SOCIETY, 1996, pp. 110–119.
19. S.M. Emran and N. Ye, *Robustness of canberra metric in computer intrusion detection*, Proceedings of the IEEE Workshop on Information Assurance and Security, West Point, NY, USA, 2001, pp. 80–84.
20. E. Eskin, *Anomaly detection over noisy data using learned probability distributions*, In Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00), 2000, pp. 255–262.

21. E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, *A geometric framework for unsupervised anomaly detection*, Applications of Data Mining in Computer Security (2002), 77–101.
22. S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, *A sense of self for unix processes*, Proceedings of the 1996 IEEE Symposium on Security and Privacy (Los Alamitos, CA), IEEE Computer Society Press, 1996, p. 120128.
23. S. Forrest, S.A. Hofmeyr, and A. Somayaji, *Computer immunology*, Communications of the ACM **40** (1997), no. 10, 88–96.
24. S. Forrest, AS Perelson, L. Allen, and R. Cherukuri, *Self-nonself discrimination in a computer*, Proceedings of the Symposium on Research in Security and Privacy, 1994, pp. 202–212.
25. AK Ghosh, J. Wanken, and F. Charron, *Detecting anomalous and unknown intrusions against programs*, Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC'98), 1998, pp. 259–267.
26. J. Gómez, D. Dasgupta, O. Nasraoui, and F. Gonzalez, *Complete expression trees for evolving fuzzy classifier systems with genetic algorithms*, Proceedings of the North American Fuzzy Information Processing Society Conference (NAFIPS-FLINTS), 2002, pp. 469–474.
27. M. Dacier H. Debar and A. Wespi, *A revised taxonomy for intrusion-detection systems*, Tech. report, IBM Research Report, 1999.
28. LT Heberlein, GV Dias, KN Levitt, B. Mukherjee, J. Wood, and D. Wolber, *A network security monitor*, Proceedings of the Symposium on Research in Security and Privacy (Oakland, CA), 1990, pp. 296–304.
29. J. Hochberg, K. Jackson, C. Stallings, JF McClary, D. DuBois, and J. Ford, *NADIR: An automated system for detecting network intrusion and misuse*, Computers and Security **12** (1993), no. 3, 235–248.
30. K. Hwang, M. Cai, Y. Chen, and M. Qin, *Hybrid intrusion detection with weighted signature generation over anomalous internet episodes*, IEEE Transactions on Dependable and Secure Computing (2007), 41–55.
31. K. Ilgun, *USTAT: A real-time intrusion detection system for UNIX*, Proceedings of the IEEE Symposium on Security and Privacy, 1993, pp. 16–28.
32. K. Ilgun, R.A. Kemmerer, and P.A. Porras, *State transition analysis: A rule-based intrusion detection approach*, IEEE transactions on software engineering **21** (1995), no. 3, 181–199.
33. KA Jackson, DH DuBois, and CA Stallings, *An expert system application for network intrusion detection*, Proceedings of the National Computer Security Conference, vol. 1, 1991.
34. Harold S. Javitz, A. Valdez, T. Lunt, and M. Tyson, *Next generation intrusion detection expert system (nides)*, Tech. Report SRI Technical Report A016, SRI International, March 1993.
35. A. Jones and R. Sielken, *Computer system intrusion detection: A survey*, Tech. report, Department of Computer Science, University of Virginia, Thornton Hall, Charlottesville, VA, September 2000.
36. C. Ko, *Logic induction of valid behavior specifications for intrusion detection*, Proceedings of IEEE Symposium on Security and Privacy, 2000, pp. 142–153.
37. Calvin Ko, Paul Brutch, Jeff Rowe, Guy Tsafnat, and Karl Levitt, *System health and intrusion monitoring using a hierarchy of constraints*, Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, (RAID 2001) (Davis, CA, USA) (W, L. M Lee, and A. Wespi, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2001, pp. 190–203.
38. Calvin Ko, Manfred Ruschitzka, and Karl Levitt, *Execution monitoring of security-critical programs in distributed systems: A specification-based approach*, Proceedings of IEEE Symposium on Security and Privacy, May 1997, pp. 175–187.
39. Christopher Kruegel and Giovanni Vigna, *Anomaly detection of web-based attacks*, Proceedings of the 10th ACM conference on Computer and communication security (Washington D.C., USA), ACM Press, October 2003, pp. 251–261.
40. S. Kumar, *Classification and detection of computer intrusions*, Ph.D. thesis, Purdue University, 1995.
41. S. Kumar and E. Spafford, *A pattern matching model for misuse intrusion detection*, Proceedings of the 17th National Computer Security Conference, 1994.

42. S. Kumar and E. Spafford, *A software architecture to support misuse intrusion detection*, Proceedings of the 18th National Information Security Conference, 1995.
43. Sandeep Kumar and Eugene Spafford, *An application of pattern matching in intrusion detection*, Tech. Report 94-013, Purdue University, Department of Computer Sciences, March 1994.
44. T. Lane, *Machine learning techniques for the computer security domain of anomaly detection*, Ph.D. thesis, Purdue University, August 2000.
45. L. Lankewicz and M. Benard, *Real-Time Anomaly Detection Using a Nonparametric Pattern Recognition Approach*, Proceedings of the 7th Annual Computer Security Applications Conference (ACSAC'91), 1991.
46. J. Lee, S. Moskovics, and L. Silacci, *A Survey of Intrusion Detection Analysis Methods*, 1999.
47. W. Lee, S. J. Stolfo, and K. W. Mok, *A data mining framework for building intrusion detection models*, Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 1999, pp. 120–132.
48. W. Lee and S.J. Stolfo, *Data mining approaches for intrusion detection*, Proceedings of the 7th USENIX Security Symposium, 1998.
49. W. Lee, S.J. Stolfo, and K.W. Mok, *Mining audit data to build intrusion detection models*, Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1998, pp. 66–72.
50. Z. Lei and A.A. Ghorbani, *Network intrusion detection using an improved competitive learning neural network*, Proceedings of the Second Annual Conference on Communication Networks and Services Research (Fredericton, NB, Canada), 2004.
51. U. Lindqvist and PA Porras, *Detecting computer and network misuse through the production-basedexpert system toolset (P-BEST)*, Proceedings of the IEEE Symposium on Security and Privacy, 1999, pp. 146–161.
52. W. Lu and I. Traore, *Unsupervised Anomaly Detection Using an Evolutionary Extension of K-means Algorithm*, International Journal on Information and Computer Security, Inderscience Publisher **2** (May, 2008), 107–139.
53. T. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D. Eclwards, P. Neumann, H. Javitz, and A. Valdes, *IDES: The Enhanced Prototype. A Real-Time Intrusion Detection System*, Tech. report, Technical Report SRI Project 4 185-010, SRI-CSL-88, 1988.
54. T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P. G. Neumann H. S. Javitz, A. Valdes, and T. D. Garvey, *A real time intrusion detection expert system (ides)*, Tech. report, SRI International, Menlo Park, CA, February 1992.
55. Teresa F. Lunt, *Detecting intruders in computer systems*, Proceedings of the 1993 Conference on Auditing and Computer Technology, 1993.
56. J. McHugh, *Intrusion and intrusion detection*, International Journal of Information Security **1** (2001), no. 1, 14–35.
57. Ludovic Me, *Gassata, a genetic algorithm as an alternative tool for security audit trails analysis*, Proceedings of the 1st International Symposium on Recent Advances in Intrusion Detection (RAID'98) (Louvain-la-Neuve, Belgium), September 1998.
58. P. G. Neumann and A. Ph. Porras, *Experience with emerald to date*, Proceedings of First USENIX Workshop on Intrusion Detection and Network Monitoring (Santa Clara, California), IEEE Computer Society Press, April 1999, pp. 73–80.
59. S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, *Modeling intrusion detection system using hybrid intelligent systems*, Journal of Network and Computer Applications **30** (2007), no. 1, 114–132.
60. J. Peng, C. Feng, and J. Rozenblit, *A hybrid intrusion detection and visualization system*, Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06), 2006, pp. 505–506.
61. A. Ph. Porras and P. G. Neumann, *Emerald: Event monitoring enabling responses to anomalous live disturbances*, Proceedings of the National Information Systems Security Conference, 1997, pp. 353–365.

62. L. Portnoy, E. Eskin, and S.J. Stolfo, *Intrusion detection with unlabeled data using clustering*, Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA'01), Philadelphia, PA, 2001, pp. 76–105.
63. M. Sabhnani and G. Serpen, *Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context*, International Conference on Machine Learning, Models, Technologies and Applications, 2003, pp. 209–215.
64. B. Scholkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson, *Estimating the support of a high-dimensional distribution*, Neural computation **13** (2001), no. 7, 1443–1471.
65. M. Sebring, E. Shellhouse, M. Hanna, and R. Whitehurst, *Expert systems in intrusion detection: A case study*, Proceedings of the 11th National Computer Security Conference, 1988, pp. 74–81.
66. R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, *Specification-based anomaly detection: a new approach for detecting network intrusions*, Proceedings of the 9th ACM conference on Computer and communication security (CCS'02) (Washington D.C., USA), ACM Press, November 2002, pp. 265–274.
67. T. Shon and J. Moon, *A hybrid machine learning approach to network anomaly detection*, Information Sciences **177** (2007), no. 18, 3799–3821.
68. M.L. Shyu, S.C. Chen, K. Sarinnapakorn, and L.W. Chang, *A Novel Anomaly Detection Scheme Based on Principal Component Classifier.*
69. V.A. Siris and F. Papagalou, *Application of anomaly detection algorithms for detecting SYN flooding attacks*, Computer Communications **29** (2006), no. 9, 1433–1442.
70. S.E. Smaha, *Haystack: An intrusion detection system*, Aerospace Computer Security Applications Conference, 1988., Fourth, 1988, pp. 37–44.
71. S. Staniford, J. Hoagland, and J. McAlerney, *Practical automated detection of stealthy portscans*, Journal of Computer Security **10** (2002), no. 1 and 2, 105–126.
72. A. Sundaram, *An introduction to intrusion detection*, Crossroads **2** (1996), no. 4, 3–7.
73. HS Teng, K. Chen, and SC Lu, *Adaptive real-time anomaly detection using inductively generatedsequential patterns*, Proceedings of the Symposium on Research in Security and Privacy (Oakland, CA), 1990, pp. 278–284.
74. J.L. Thames, R. Abler, and A. Saad, *Hybrid intelligent systems for network security*, Proceedings of the 44th annual Southeast regional conference, ACM New York, NY, USA, 2006, pp. 286–289.
75. Marina Thottan and Chuanyi Ji, *Anomaly detection in ip networks*, IEEE Transactions on Signal Processing **51** (2003), no. 8, 148–166.
76. E. Tombini, H. Debar, L. Me, M. Ducasse, F. Telecom, and F. Caen, *A serial combination of anomaly and misuse IDSes applied to HTTP traffic*, Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), 2004, pp. 428–437.
77. Prem Uppuluri and R. Sekar, *Experiences with specification-based intrusion detection*, Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, (RAID 2001) (Davis, CA, USA) (W, L. M Lee, and A. Wespi, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2001, pp. 172–189.
78. H. S. Vaccaro and G. E. Liepins, *Detection of anomalous computer session activity*, Proceedings of the Symposium on Research in Security and Privacy (Oakland, CA), May 1989, pp. 280–289.
79. A. Valdes and K. Skinner, *Adaptive, model-based monitoring for cyber attack detection*, Lecture Notes in Computer Science (2000), 80–92.
80. G. Vigna, S.T. Eckmann, and R.A. Kemmerer, *The stat tool suite*, Proceedings of DISCEX 2000 (Hilton Head, SC), IEEE Press, January 2000, pp. 46–55.
81. G. Vigna and RA Kemmerer, *NetSTAT: A network-based intrusion detection approach*, Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC'98), 1998, pp. 25–34.
82. G. Vigna and R.A. Kemmerer, *NetSTAT: A network-based intrusion detection system*, Journal of Computer Security **7** (1999), no. 1, 37–71.

83. G. Vigna, W. Robertson, V. Kher, and R.A. Kemmerer, *A stateful intrusion detection system for world-wide web servers*, Proceedings of the Annual Computer Security Applications Conference (ACSAC 2003) (Las Vegas, NV), December 2003, pp. 34–43.
84. G. Vigna, F. Valeur, and R.A. Kemmerer, *Designing and implementing a family of intrusion detection systems*, Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003) (Helsinki, Finland), September 2003.
85. C. Warrender, S. Forrest, and B. Pearlmutter, *Detecting intrusions using system calls: alternative data models*, Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 1999, pp. 133–145.
86. C. Xiang and S.M. Lim, *Design of multiple-level hybrid classifier for intrusion detection system*, Proceedings of the 2005 IEEE Workshop on Machine Learning for Signal Processing, 2005, pp. 117–122.
87. A.A. Ghorbani Y. Guan and N. Belacel, *Y-means : A clustering method for intrusion detection*, IEEE Canadian Conference on Electrical and Computer Engineering, Proceedings, 2003.
88. K. Yamanishi, J.I. Takeuchi, G. Williams, and P. Milne, *On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms*, Data Mining and Knowledge Discovery **8** (2004), no. 3, 275–300.
89. B. Yu, E. Byres, and C. Howey, *Monitoring Controller's" DNA Sequence" For System Security*, ISA Emerging Technologies Conference, Instrumentation Systems and Automation Society, 2001.
90. J. Zhang and M. Zulkernine, *A hybrid network intrusion detection technique using random forests*, The First International Conference on Availability, Reliability and Security (ARES'06), 2006, pp. 262–269.