1 Outlier Detection

Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu

CONTENTS

1.1 INTRODUCTION

Outlier detection is an important data analysis task. It is used in many domains to identify interesting and emerging patterns, trends, and anomalies from data. Outlier detection is used to detect anomalies in many different domains, including computer network intrusion; gene expression analysis; disease onset identification, including cancer detection; financial fraud detection; and human behavioral analysis. Among the four primary tasks of data mining, outlier detection is the closest to the motivation of data mining as discovering interesting patterns and modeling relationships are the main aims of data mining research. Outlier detection, also known as anomaly detection, deviation detection, novelty detection, and exception mining, has been widely studied in data mining as well as in statistics and machine learning.

An outlier is a special event or object that is not similar to the rest of the data. Outliers are considered to be important because they may represent significant information that often requires critical actions to be taken in a wide range of application domains. For example, an outlier in a MRI image may indicate the presence of a malignant tumor. Abnormal behaviors in credit card transactions could indicate forgery, and an unusual traffic pattern in a network could mean that a computer is hacked and it is transmitting classified data to an unauthorized destination.

Computer intrusion includes hacking and spreading of viruses and worms across networks to infiltrate a local or remote machine or to cause damage using distributed denial of service (DDoS) attacks. However, an intrusion constitutes only a small percentage of the total network and computer usage that is considered normal usage. This small number of intrusion activities is very different from normal or regular user activities and hence can easily be detected using outlier detection techniques. Outlier detection can be used to identify malicious activities of programs as well as hackers from network traffic data and computer activities.

In this chapter, outlier detection has been introduced along with case studies and explanations of different outlier detection techniques. Existing outlier detection techniques based on various application domains have been discussed. This chapter intends to be a comprehensive reference in the field of intrusion detection and prevention. It can be used as a reference for academicians and also as a suitable textbook for final-year undergraduate and postgraduate students.

We begin in Section 1.2 with the basics of outlier detection. We provide definitions of "outlier" widely adapted in the field. Then we discuss the relationship with anomaly detection, data types, outlier types, evaluation methods, and research challenges.

In Section 1.3, we highlight the data labels and how outlier detection techniques are classified based on data labels. We also mention the output of outlier detection.

Section 1.4 includes the application domain of outlier detection. We discuss few important domains and have included one case study for intrusion detection.

In Section 1.5, we discuss clustering-based outlier detection methodologies. These techniques are classified into two groups based on their output. These are scoring-based and binary-based approaches. We also discuss the merits and demerits of these techniques.

In Section 1.6, statistical-based approaches for outlier detection are discussed in two broad categories, including their advantages and disadvantages. Finally, we conclude the chapter in Section 1.7 followed by necessary references.

1.2 BASICS OF OUTLIER DETECTION

1.2.1 Definition

Out of four primary tasks of data mining, outlier detection is the closest to the motivation of data mining for discovering interesting patterns. Outlier detection has been widely researched from not only the data mining perspective, but also in statistics and machine learning [1]. One widely accepted definition of "outlier" is given by Hawkins [2]. According to him, "An outlier is an observation

which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism." Barnett and Lewis [3] defined outlier: "An outlier is an observation or subset of observations which appears to be inconsistent with the remainder of data." Outlier detection is also defined specifically for different methods such as the following:

Clustering-based outlier: Outliers are the points that do not belong to clusters of a data set or as clusters that are significantly smaller than other clusters [4,5].
Depth-based outlier: Outliers are the points in the shallow convex hull layers with the lowest depth [6].
Graph-based outlier: Outliers are the points that are present in particular positions in the graph [7].
Density-based outliers: Outliers are the points that lie in the lower local density with respect to the density of its local neighborhood [8].
Neural network–based outlier: Points that are not reproduced well at the output layer with high reconstruction error are considered as outliers [9].
Support vector machine–based outlier: Points that are distant from most other points or points that are present in relatively sparse regions of the feature space are considered as outliers [10].
Spatial outlier: A spatial outlier is a spatially referenced point whose non-spatial attribute values are significantly different from those of other spatially referenced points in its spatial neighborhood [11].

1.2.2 Relationship with Anomaly Detection

Outlier detection is also named as anomaly detection, deviation detection, novelty detection, exception mining, etc. Outlier detection and anomaly detection are almost the same, but outliers are more focused on data and follow strict rules, for example, in normally distributed data, $\mu + 3\sigma$ is considered as an outlier. Anomalous events have no objection to fall under rules. For example, one goes to play lawn tennis every afternoon in a month, but if the same person plays table tennis one afternoon instead of lawn tennis, that can be considered as an anomalous event. If we consider this event in a data set, then this can be detected as an outlier. So, anomalies and outliers are related to each other and hence can be researched together.

1.2.3 Data Types

One important issue of outlier detection is the nature of data. In outlier detection, perspective data is also regarded as a point, object, record, vector, sample, observation, etc. Each data instance can be explained with a group of features or attributes. Basically, the attributes form a data instance and its characteristics. Attributes can be binary, continuous, or categorical. Each data instance can be comprised of different types of attributes. Generally, there are two types of data, simple and complex.

1.2.3.1 Simple Data

Simple data are the most commonly used data with low dimensionality and real-valued attributes. Almost all outlier detection techniques can be applied to simple data.

1.2.3.2 Complex Data

Complex data creates significant challenges for outlier detection techniques due to their nature. Complex data has a mixture of different types of attributes with high dimensions. Sequence data, spatial data, streaming data, and spatio-temporal data are generally considered to be complex.

1.2.4 Types of Outliers

Based on data and detection techniques, outliers can be categorized into the following groups:

1.2.4.1 Point Outliers

When a particular data instance does not follow the rest of the data, it can be considered as a point outlier. For a realistic example, we can consider expenditure on car fuel. If usual fuel consumption per day is five liters and one day it becomes 50 liters, then obviously it is a point outlier. Figure 1.1 elucidates the point outlier concept.

1.2.4.2 Contextual Outliers

When a data instance is behaving anomalously in a particular context and not otherwise, then it is termed as a contextual outlier, which can also be referred to as a conditional outlier. For example, the expenditure on a credit card during a festive period is not like at a regular time of the year. If the regular expenditure every month is $500, which is normal, then an expenditure of $2000 during a nonfestive period is considered abnormal and hence is a contextual outlier (Figure 1.2).

1.2.4.3 Collective Outliers

When a collection of similar data instances are behaving anomalously, then with respect to the entire data set that collection is termed a collective outlier. It might happen that the individual data instance is not an outlier by itself, but due to its presence in a collection, it is becoming an outlier.

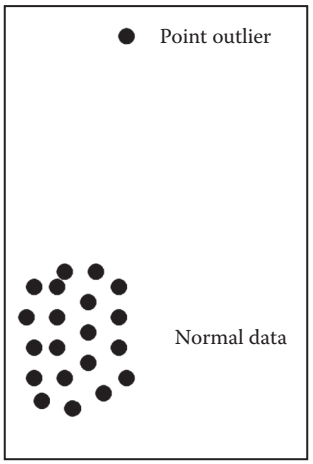

FIGURE 1.1 Point outlier.

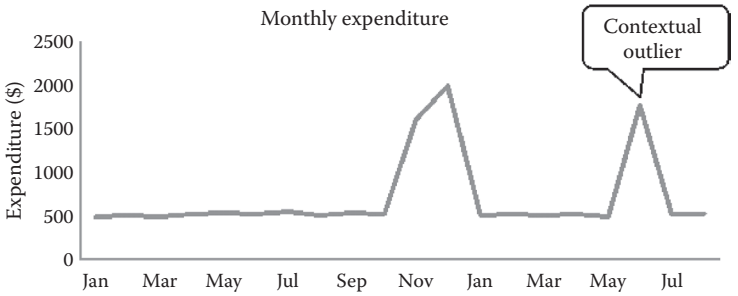

FIGURE 1.2 Contextual outlier.

For example, in a human electrocardiogram output, the existence of a low value for a long period of time indicates an outlying phenomenon corresponding to abnormal premature contraction [12]. But one low value itself is not an outlier.

1.2.5 Evaluation Methods

Outlier detection techniques are evaluated mainly in the following ways:

1.2.5.1 ROC Curve

A receiver operating characteristic (ROC) curve [13] is a 2-D graph used to show detection rates and false alarm rates. Detection rate defines the correctly identified outliers whereas a false alarm rate is the opposite. Figure 1.3 represents an ideal ROC curve in which the detection rate is high, and the false alarm rate is very low.

1.2.5.2 Computational Complexity

Efficiency of outlier detection methods can be also computed by computational cost, which seems to be a universal method. An efficient technique is reflected on its time and space complexity regardless of type of data. Moreover, efficient techniques should have the ability to handle large and high dimensional data with a lower computational burden.

1.2.5.3 User-Defined Parameters

Existing outlier detection techniques require user-defined parameters, which are very critical for effectiveness and efficiency. The optimal parameter choice is not always easy and requires extensive testing. Therefore, the minimal use of user-defined parameters is more applicable for outlier detection methods.

1.2.6 Research Challenges

Although outlier detection seems to be very straightforward as we just have to find the data that do not follow the normal behavioral pattern. But there are still some research challenges regardless of so many techniques available. Research challenges are as follows:

a) No universal outlier detection technique is available. One technique in a domain is not suitable for others. For example, in the medical domain, outliers are very subtle whereas in image processing it might be considered as a normal phenomenon.
b) Data contains noise, which tends to be anomalous and hence difficult to segregate.
c) Labeled data for training of models used by many techniques are hardly available.

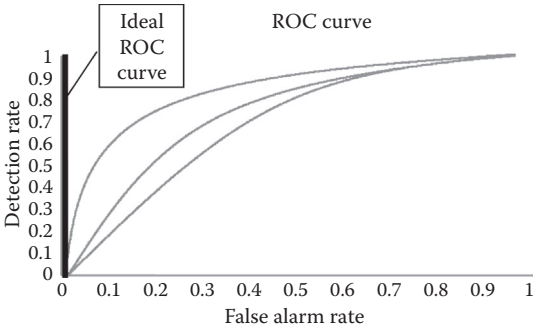

FIGURE 1.3 ROC curves.

d) Normal behaviors keep evolving; hence normal behavior once is not normal forever. Thus current outlier detection techniques might be of no use in the future.
e) When outliers are caused by fraudulent activities, the malicious adversaries try to make abnormal activities appear normal; thus such behaviors are difficult to detect.

For these above-mentioned challenges, outlier detection is not an easy problem to solve. Most of the existing techniques are based on application domain and the nature of data. Researchers from diversified disciplines have adopted miscellaneous concepts to solve these problems.

1.3 DATA LABELS

Data labels associated with each data instance specify the status of the data, i.e., normal or abnormal. We discussed in Section 1.6 about the challenges, and we mentioned it is very expensive to obtain labeled data. Data labeling is handled by human experts or analysts, and hence it becomes a difficult job to get prelabeled data for outlier detection. Moreover, some outliers cannot be labeled because of dynamic characteristics. Based on labeling, outlier detection is categorized into three of the following categories:

1.3.1 Supervised Outlier Detection

These approaches require the learning of normal and outlying models from the knowledge of prelabeled data. According to this learning, supervised techniques detect a new data instance as normal or outlying. Actually it is tested with the normal and outlying models and checks where it is fitted. Supervised approaches are basically used in intrusion detection, but gaining access to prelabeled data is the main challenge of these approaches.

1.3.2 Unsupervised Outlier Detection

These approaches do not require any prelabeled data to detect outliers. For example, in normally distributed data, we use the three-sigma rule to detect outliers. Some methodologies use distance to detect outliers without any prelabeled data. Compared with supervised approaches, unsupervised approaches are more applicable in various real life domains.

1.3.3 Semisupervised Outlier Detection

These approaches require prelabeled data but only normal class. These approaches do not require the outlying class data to be labeled beforehand, thus these approaches can be widely applicable and more usable than supervised techniques. There are many situations in which outlying scenarios are difficult to model, like in spacecraft fault detection [14] in which accidents are outlying events, which are difficult to model. So, semisupervised approaches detect outliers by comparing new data points with the prelabeled normal class.

1.3.4 Output of Outlier Detection

One important issue in outlier detection is how the outliers are represented as output. Generally, there are these two following categories:

Scores: Scoring-based outlier detection techniques assign an outlier score to each of the data instances. Then the scores are ranked, and an analyst uses them to decide the outliers or use a threshold to select outliers.
Binary: According to these techniques, outputs are considered in a binary fashion, i.e., either outlier or nonoutlier.

Techniques that provide binary labels are computationally efficient because each of the data instances do not have to provide an outlier score.

1.4 APPLICATION DOMAINS

Outlier detection has been widely applied in various application domains. We highlight a few important domains here.

1.4.1 Intrusion Detection

Computer intrusion includes hacking and spreading of viruses and worms across networks to infiltrate a local or remote machine or cause damage using distributed denial of service (DDoS) attacks. However, intrusions constitute only a small percentage of the total network and computer usage. Outlier detection can be used to identify malicious activities of programs as well as hackers [15] from network traffic data and computer activities. Outlier detection techniques need to be computationally efficient to handle large-sized input data, which also causes a false alarm rate. The data that come in a streaming fashion requires online analysis. IDS are classified into two groups. These are host based and network based. The major differences between them are in the nature of outliers, profiling, and data analysis.

1.4.1.1 Case Study: Haystack

The Haystack prototype [16] was developed for intrusion detection in a multiuser US Air Force computer system. It was a Unisys 1100/60 mainframe computer with an OS/1100 operating system. Haystack was designed to identify six different types of intrusions:

a) Malicious use: Sundry attacks like deletion of files, resource misuse, etc.
b) Denial of service: Keeping the resource unavailable to other users.
c) Penetration of security control systems: Attempts to handle the security characteristics of the system by a user.
d) Leakage: Transferring important data from the system.
e) Masquerade attacks: When an unauthorized user attempts to assume the identity of another user.
f) Attempted break-ins: When an unauthorized user wants to access the computer illegally.

To detect these six types of intrusions, the system uses outlier detection techniques. For Haystack, it is supervised outlier detection as the past behaviors are modeled using prelabeled data.

1.4.2 Fraud Detection

Fraud detection considers the criminal activities in commercial organizations. The most vulnerable areas of fraudulent activities are credit cards, cell phones, insurance claims, and insider trading. Stolen credit cards are used in a more unusual way than the normal pattern. The usage behavior of the stolen credit card is not like that of the regular user. The identification of such activities helps prevent thieves and reduces the monetary loss of the individual [17]. To prevent the misuse of a mobile phone account, it is necessary to detect the unusual usage pattern. The basic function is to monitor the calling behavior of each account and issue an alarm when an account appears to be misused. Automobile insurance fraud is a very common crime. Criminal rings of illegal claimants and providers manipulate the claim processing system for unauthorized claims. Tracking such activities helps the company to avoid financial losses. Neural network–based techniques are applied to detect such activities [18,19]. In recent times, the stock market business attracted enough mass people. Insider trading is a criminal activity in business, in which profit is made by inside information before it is made public [20].

1.4.3 Medical and Public Health

Anomalous records can be generated due to patient condition or instrumental error or recording errors. A wrong test report might have serious repercussions. On the other hand, outlier detection in this domain is a very important tool that can potentially save human lives [21] by detecting a problem early from test results and images. The most challenging aspect of the outlier detection problem is that the cost of classifying an outlier as normal can be very high. Outlier detection in this domain requires maximum accuracy. The data consists of different attributes of a patient, such as age, blood pressure, weight, and height. Most of the techniques use a semisupervised approach. Collective anomaly detection techniques have been used to detect anomalies in time series data, such as electrocardiograms (ECGs) and electroencephalograms (EEGs).

1.4.4 Sensor Networks

A sensor network might comprise a multiple number of sensors in which each sensor collects different types of data. The collected data often contains noise and missing values due to limitations created by environmental and communication channels. Outlier detection in a sensor network has several real-life applications, such as environmental monitoring, habitat monitoring, industrial monitoring, target tracking, surveillance monitoring, etc. Conventional outlier detection techniques might not be suitable for handling sensor data for the following reasons: resource constraints, high communication costs, distributed streaming, dynamic network topology, large-scale deployment, etc. [22].

1.4.5 Image Processing

This domain includes satellite imagery [23], digit recognition [24], and mammographic image analysis [25]. The outliers are generated by motion or instrumental faults. The data has spatial as well as temporal characteristics. Each data point has continuous attributes like color, lightness, texture, etc. Interesting outliers are supposed to be anomalous points or a particular region of the image. The main challenge of this domain is to deal with large-size input data.

The above-mentioned domains are not the end. Outlier detection is also applied to several other domains like astronomical data [26], biological data [27], speech recognition [28], robot behavior [29], and many more.

1.5 CLUSTERING-BASED OUTLIER DETECTION TECHNIQUES

Clustering is a way of grouping similar objects in a group. Using clustering techniques, many outlier detection techniques have been proposed by various researchers. Here in this section, we classify these techniques based on output.

1.5.1 Binary Output

Knorr and Ng et al. [30] presented the algorithms to detect distance-based outliers. They consider a data point O in a data set T, a $DB(p;D)$ outlier, if at least a fraction p of the data points in T lies greater than distance D from O. Their index-based algorithm executes a range search with radius D for each data point. If the number of data points in its D neighborhood exceeds a threshold, the search stops, and that data point is declared as a nonoutlier; otherwise, it is an outlier. In their cell-based approach, they quantize the complete data space and assign the data points to the cells. By pruning away a large number of red cells that contain too many data points and their immediate neighbors, their approach avoids testing unnecessary cells and speeds up outlier detection. Their experiments show that their cell-based algorithm is the most efficient when the number of

dimensions is less than or equal to four. However, for a higher number of dimensions (>5), the number of cells grows exponentially, and the nested loop that they provided in the same paper outperforms the cell-based algorithm.

Yang et al. [31] attempted to automatically detect novel events from a temporally ordered stream of news stories, either retrospectively or as the stories arrive. The objective is to identify stories in several continuous news streams that belong to previously unidentified events. This can be done in an online fashion, i.e., as the events occur or as an accumulated collection. In retrospective event detection, stories are grouped together, and each cluster uniquely identifies an event. In online event detection, each document is labeled, and a single pass algorithm (INCR) generates a nonhierarchical partition of the input collection. The former is appropriate for retrospective event detection whereas the latter can be used for both. A story is represented using a vector of weighted terms. The normalized vector sum of documents in a cluster is used to represent the cluster, and it is called a prototype or centroid. The standard cosine similarity measure is used to describe the similarity of a cluster centroid and a document. GAC is an agglomerative algorithm that maximizes the average similarity between document pairs in the resulting clusters. At each iteration, it divides the current set of clusters into buckets and does local clustering within each bucket. The process is repeated and generates clusters at higher and higher levels until a predefined number of clusters are obtained. The input to the algorithm is a set of documents, and the output is a forest of cluster trees with the number of trees specified by the user. Clusters are produced by growing a binary tree using the bottom-up approach. Novelty detection is used in the case of single-pass clustering. The algorithm sequentially processes the input documents, one at a time, and grows clusters incrementally. A new document is classified to its most similar cluster if the similarity exceeds a predefined threshold; otherwise, it becomes the seed for a new cluster. By adjusting the threshold, one can obtain clusters at different levels of granularity.

David et al. [32] proposed clustering based on multivariate outlier detection by using Mahalanobis distance. At first, calculate the Mahalanobis distance for n observations on P variables. Then determine the observations that are above the upper control limit (UCL) of the T-square statistic [33] and consider those observations to be outlier cluster 1. Repeat the same procedure until the nature of the variance-covariance matrix for the variables in the last cluster achieves singularity. Their experimental data was comprised of 19 different attributes about a two-wheeler company in India, and data was collected from 275 two-wheeler users.

$$(\text{Mahalanobis distance})_i = \sqrt{(X_i - X_{mean})^T S^{-1} (X_i - X_{mean})} \tag{1.1}$$

$$T_i^2 = (X_i - X_{mean})^T S^{-1} (X_i - X_{mean}) \tag{1.2}$$

Zoubi et al. [34] proposed fuzzy clustering-based outlier detection. At first they execute a fuzzy clustering algorithm [35] to produce a set of k clusters and objective function and calculate a threshold from the average distances between each point in the cluster. Then they determine the small cluster and consider them as outliers. For the rest of the data, they remove a point and recalculate the objective function. If the change in objective function is greater than the threshold, then the point is considered as an outlier. They worked with iris data and bupa data from a UCI machine-learning repository [36]. Their approach of calculating the threshold from average has no significant explanation and hence might result in poor performance in large-scale high-dimensional data sets. The objective function for FCM is

$$J(U, c_1, \ldots c_c) = \sum_{i=1}^{c} J_i = \sum_{i=j}^{c} \sum_{j}^{n} u_{ij}^m d_{ij}^2 \tag{1.3}$$

Another clustering-based approach by Zoubi et al. [39] used partition around medoid (PAM), a data order–independent algorithm, to provide a set of clusters and a set of medoids (cluster center). Small clusters are then determined as outliers. The definition of a small cluster includes the clusters with fewer points than half of the average number of points in k clusters. There is no justification of the calculation or any proof for this small cluster determination. Then, to detect outliers in the rest of the data, a threshold value is calculated from the average of absolute distances between medoid and each point of each cluster. If a particular distance is greater than the threshold, it is considered as an outlier. Their experimental data was iris data and bupa data [36].

Yoon et al. [37] used k-means clustering on the complete data set, and to find the proper value of k, used cubic clustering criterion (CCC). CCC is a technique used to estimate the number of clusters evaluated by the Monte Carlo method. Once the clustering is done, the domain expert searches for external and internal outliers in the clustering results. External outliers are the data points positioned at a greater distance than other clusters, and internal outliers are the data points distantly positioned inside a cluster. If the removal of outliers creates meaningful clusters, then the procedure halts. Their approach has been applied only for software measurement data, and a domain expert is required for interpretation.

Loureior et al. [38] used hierarchical clustering to detect outliers. Their key idea is to use the size of resulting clusters as indicators of the presence of outliers. As their methodology is tested on the problem cleaning official statistics data and the goal is to detect erroneous foreign trade transaction in data collected by the Portuguese Institute of Statistics (INE), the outliers are the unusual values that are distant from the normal and more frequent observations and that, therefore, will be isolated in smaller clusters.

Jiang et al. [5] presented a two-phase clustering technique to detect outliers. First, they used a modified k-means algorithm to create clusters. If the points in the same cluster are not close enough, the cluster can be split into two smaller clusters and merged when a given threshold is exceeded. In the second step, they construct a minimum spanning tree with cluster centers and remove the longest edge. The smaller subtrees are considered as outliers. Their technique considers an entire cluster as an outlier, which may not be applicable for many data sets and increase the false positive rate.

Yu et al. [4] presented an approach, FindOut. Using the multi-resolution property of wavelet transforms, FindOut can successfully identify various percentages of outliers from large data sets. It can also detect the outliers for complicated data patterns with various densities. Furthermore, FindOut can handle high-dimensional data sets, which have not been addressed by most existing approaches.

Given a linear space system LSw and density function φ, a data point in a cell c is an outlier if $\hat{} w(c) < \tau$, where τ is given. For a data set, it defines a density function φ. Kernel function g is defined as follows:

$$G(c_i) = \begin{cases} 1 \text{ if } D(c_i, \underset{0}{\rightarrow}) \leq \epsilon \\ 0 \quad \text{otherwise} \end{cases} \tag{1.4}$$

where $\underset{0}{\rightarrow}$ is the origin in the data space, and D is a distance function. A linear system LSg is defined based on g. A data point oi in LSg is an outlier if $\hat{} g(oi) < p \times N$, where N is the size of the data points.

1.5.2 Scoring Output

He et al. [18] proposed another definition for cluster-based local outliers. According to their definition, all the data points in a certain cluster are considered as outliers rather than a single point as shown in Figure 1.4. The smaller clusters C1 and C3 are considered as outliers. They used some numeric

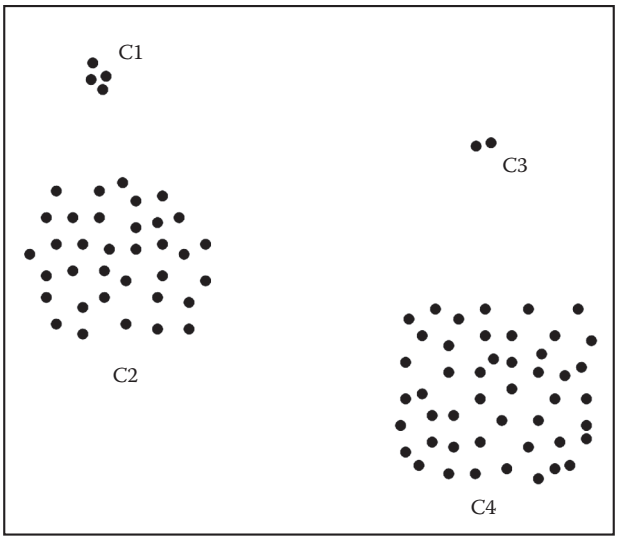

FIGURE 1.4 Cluster-based outlier. (From He, Z. et al., *Pattern Recognition Letters, 24,* 9–10, 1641, 2003.)

parameters, i.e., a and b, to identify small clusters (SC) and large clusters (LC). The clustering technique depends on these parameters, but it is not clear how the values can be set for various data sets. They used the SQUEEZER algorithm to cluster data as it achieves both a high quality of clustering and can handle high-dimensional data. Then, the FindCBLOF algorithm determines the outlier factor of each individual record in the data set. CBLOF(t) for each record t is calculated as follows:

$$\text{CBLOF}(t) = \begin{cases} |C_i| * \min\left(d(t, C_j)\right) \text{where } t \in C_i, C_i \in SC \text{ and } C_j \in LC \text{ for } j = 1 \text{ to } b \\ |C_i| * \left(d(t, C_i)\right) \text{where } t \in C_i \text{ and } C_i \in LC \end{cases} \tag{1.5}$$

Svetlona et al. [40] presented an outlier removal clustering (ORC) algorithm that provides outlier detection and data clustering simultaneously. Their proposed algorithm has two stages. At first the *k-means* clustering is applied and then an *outlyingness factor*, o_i, for each of the data points is calculated by taking the ratio of a point's distance to the centroid and the maximum distance from centroid to any other point. A threshold T is set less than one to check for outliers. If the outlying factor for any point is greater than the threshold, then it is considered as an outlier and removed from the data set. Their experimental data includes synthetic data and some map images. Mean absolute error (MAE) is used to evaluate algorithm performance. The parameter T value is dependent on the data set, which may cause poor performance in heterogeneous large-scale data sets.

$$o_i = \frac{x_i - C_{pi}}{d_{max}} \tag{1.6}$$

Ren et al. [41] proposed a vertical outlier detection algorithm with clusters as a byproduct. Their approach doesn't require beforehand clustering of the data; rather it is a one-time process. Outliers are measured by a local connective factor (LCF), which indicates how significantly the point connects with other points in a data set. LCF is calculated by a vertical data representation P tree. Outliers are the points that are not connected with clusters. They defined a neighborhood density factor to calculate the LCF of each point in the data set. Their experimental analysis was based on run times and scalability to data size and compared with two-phase clustering [5] and a cluster-based local outlier [18].

The LCF of the point *P*, denoted as LCF(P,r) is the ratio of $DF_{nbr}(P,r)$ over $DF_{cluster}(P,R)$.

$$\mathrm{LCF(P,r)} = \frac{\mathrm{DF_{nbr}(P,r)}}{\mathrm{DF_{cluster}(P,R)}} \tag{1.7}$$

$Ol_{factor} = N(Nbr(P,r))*DF_{nbr}(P,r)$, where *r* is the radius of neighborhood of *P*.
The outlier set is denoted as

$$\mathrm{Ols}(X, t) = \{x \in X \mid Ol_{factor}(x) < t\} \tag{1.8}$$

Fan et al. [42] introduced a nonparametric outlier detection technique for efficiently discovering top-*n* outliers from engineering data (2-D and 3-D synthetic data). Their algorithm generates reasonable outlier results by taking both local and global features of a data set into consideration. They proposed a resolution-based outlier factor (ROF) as

$$\mathrm{ROF(O)} = \sum_{i=1}^{R} \frac{ClusterSize(O, r_{i-1}) - 1}{ClusterSize(O, r_i)} \tag{1.9}$$

where $r_0,.....,r_R$ is the resolution at each step. A resolution-based outlier mining algorithm works in two steps. At first, the RB-CLUSTER algorithm is used to cluster and label all the objects. Then the RB-MINE algorithm ranks the ROF values in an increasing order and obtains top-*N* outliers. The RB-outlier measures an object against its degree of outlyingness by taking both "global" and "local" features into account.

Breunig et al. [8] proposed an idea to assign each object a degree of being an outlier. This degree is called the local outlier factor (LOF). LOF depends on how isolated the object is with respect to the surrounding neighborhood. The local outlier factor of an object *p* is

$$LOF^{(p)}_{MinPts} = \frac{\sum_{o \in N^{(p)}_{MinPts}} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{\left|N_{MinPts}(p)\right|} \tag{1.10}$$

This outlier factor of object *p* calculates the degree to which *p* can be called as an outlier. The outlier factor is the average of the ratio of the local reachability density of *p* and those of *p*s MinPts–nearest neighbors. The author also described mathematically the LOF for objects deep in a cluster along with general bounds (upper, lower, and tight). The impact of MinPts to calculate LOF is also elaborated with necessary examples. Their approach can intelligently choose the range of *k*; the LOF approach has a lower computational complexity than the depth-based approaches for large dimensionality.

Jin et al. [43] proposed an approach for mining only top-*n* local outliers because the LOF [45] values for every data object require a large number of *k*–nearest neighbor searches and can be very much computationally expensive. They proposed an efficient microcluster-based local outlier mining algorithm to find the top-*n* local outliers in a large database. A microcluster MC (*n*, *c*, and *r*) is a summarized representation of a group of data $p_1,\ldots,p_n$, which are so close together that they are likely to belong to the same cluster. Here, $c = \frac{\sum_{i=1}^{n} p_i}{n}$, is the mean center while $r = \max\{d(p_i,c)\}$, $i = 1,...,n$, is the radius. Data are compressed into small clusters, and small clusters are represented using some statistical information as microclusters. Three different algorithms are used to find top-*n* local outliers. At first, *k*-distance bounds for each microcluster are computed. Then using these *k*-distance bounds, the LOF bound is calculated. Finally given an upper bound and a lower bound for the LOF of each microcluster, top-*n* local outliers are ranked.

Ramaswamy et al. [44] provided outlier definition based on the distance of a point from its *k*th nearest neighbor. They provided a ranking of top-*n* outliers by the measure of the outlierness of the points. According to them, top-*n* points with the maximum distance to their own *k*th nearest neighbor are considered as outliers. They also exploited index-based and nested-loop algorithms to detect outliers. Furthermore, they proposed a partition-based algorithm to prune and process the partitioned groups to improve efficiency for outlier detection. Their algorithm reduces the cost of computation in large, multidimensional data sets.

He et al. [45] introduced a new definition for outlier, the semantic outlier. A semantic outlier is a data point that behaves differently from the other data points in the same class. A measure for identifying the degree of each object being an outlier is presented, which is called the semantic outlier factor (SOF). To mine semantic outliers, an algorithm is also proposed. They used a SQUEEZER algorithm, which is used to produce good clusters for categorical data sets and then used their algorithm to calculate the SOF value for each of the objects. Their proposed outlier definition works by identifying the similarity between a specific set and a record. Given a set of records *R* and a record *t*, the similarity between *R* and *t* is defined as follows:

$$\mathrm{Sim(t,R)} = \frac{\sum_{i=1}^{|R|} similarity(t, T_i)}{|R|} \quad \text{where } \forall\ T_i \in R \tag{1.11}$$

The semantic outlier factor of a record *t* is defined as:

$$\mathrm{SOF}(t) = \frac{\Pr(cl_i|C_k) * sim(t,R)}{\Pr(cl_i|D)} \tag{1.12}$$

Spiros et al. [46] introduced local correlation integral (LOCI) for evaluating outlierness, which is very efficient in detecting outliers and groups of outliers. The main advantage of this approach is an automatic data-dictated cut-off to determine whether a point is an outlier. They introduced the multigranularity deviation factor (MDEF), which at radius *r* for a point p_i is the relative deviation of its local neighborhood density from the average local neighborhood density in its neighborhood.

$$\mathrm{MDEF}(p_i, r, \alpha) = 1 - \frac{n(p_i, \alpha r)}{\hat{n}(p_i, \alpha, r)} \tag{1.13}$$

A point is flagged as an outlier if for any $r \in [r_{\min}, r_{\max}]$ its MDEF is sufficiently large, i.e.,

$$\mathrm{MDEF}(p_i, r, \alpha) > k_\sigma \sigma_{MDEF}(p_i, r, \alpha) \tag{1.14}$$

Zhange et al. [47] proposed a new outlier detection definition, local distance-based outlier factor (LDOF), which is sensitive to outliers in scattered data sets. LDOF uses the relative distances from an object to its neighborhood to measure how much objects deviate from their scattered neighborhood. The higher the violation degree an object has, the more likely the object is an outlier. The local distance-based outlier factor of x_p is defined as

$$\mathrm{LDOF}_k(x_p) = \frac{\bar{d}_{x_p}}{\bar{D}_{x_p}} \tag{1.15}$$

where $\bar{d}_{x_p}$ the *k*-nearest neighbors are the distance of object x_p and $\bar{D}_{x_p}$ is the *k*-nearest neighbor's inner distance of x_p (Figure 1.5).

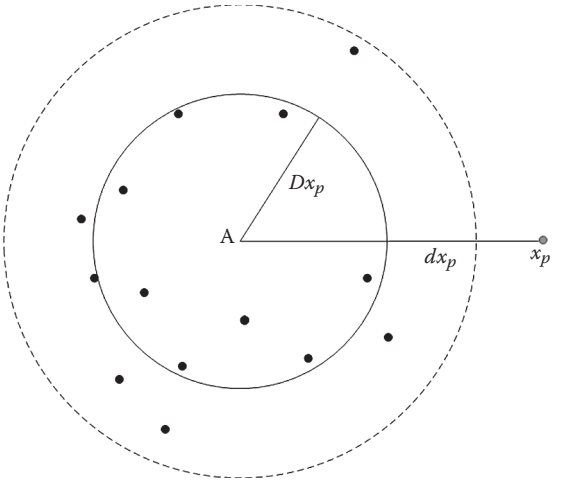

FIGURE 1.5 The explicit outlierness of object x_p with the help of LDOF definition. A is the center of the neighborhood system of x_p. The dashed circle includes all neighbors of x_p. The solid circle is x_p's "reformed" neighborhood region. (From Zhang, K. et al., In *Proc. PAKDD*, 2009.)

Kriegel et al. [48] formulated a local density-based outlier detection method providing an outlier "score" in the range of [0, 1] that is directly interpretable as a probability of a data object for being an outlier. The probabilistic local outlier factor (PLOF) of an object $o \in D$ w.r.t. a significance, a context set $S(o)$, can be defined as follows:

$$PLOF_{\lambda,S}(o) = \frac{pdist(\lambda, o, S(o))}{E_{s\in S(o)}[pdist(\lambda, s, S(s))]} - 1. \tag{1.16}$$

To achieve a normalization making the scaling of PLOF independent of the particular data distribution, the aggregate value nPLOF is obtained during PLOF computation.

$$\text{nPLOF} = \lambda.\sqrt{E[(PLOF)^2]} \tag{1.17}$$

Local outlier probability (LoOP), indicating the probability that a point $o \in D$ is an outlier:

$$\text{LoOP}_s(\text{o}) = \max\left\{0, erf\left(\frac{PLOF_{\lambda,s}(o)}{nPLOF.\sqrt{2}}\right)\right\} \tag{1.18}$$

1.5.3 Advantages and Disadvantages

The techniques used to detect outliers in binary fashion are computationally efficient irrespective of the clustering algorithm because each object in a data set is not required to assign an outlying factor as with scoring-based output. The top-N outlier concept is absent in these techniques and hence is unsupervised. The main drawback of these techniques is the accuracy of detecting all the rare class instances; because all the data objects are not taken into consideration for being outliers, many of them might be missing, and normal instances may be detected as outliers.

The scoring-based techniques have the maximum efficiency in detecting outlier accurately because all the objects are under consideration as candidate outliers. But the loophole of these techniques is computational cost because all the objects are taken under consideration to assign the outlyingness factor. Top-*N* outliers must have to be specified by a data analyst, and thus the approach becomes supervised.

1.6 STATISTICAL-BASED APPROACHES

These approaches are the first-generation techniques for outlier detection. Actually, these techniques are called model-based techniques. Models are based on probability distribution of the data, and outliers are detected as to how well the data fit into the model. Statistically based approaches are categorized into two groups, depending on probability distribution, as follows:

1.6.1 Parametric Approaches

In these approaches, the probability distribution of the data is known. Then, using the distribution parameters, outliers are detected. A point is an outlier if it deviates significantly from the data model; however, in many situations, a prior knowledge of distribution is not possible to attain.

Wu et al. [49] proposed two algorithms for outlying sensors and event boundary detection. The basic idea of outlying sensor detection is as such, each sensor first computes the difference between its reading and the median reading from the neighboring reading. Each sensor then collects all differences from its neighborhood and standardizes them. A sensor is an outlier if the absolute value of its standardized difference is sufficiently large. The algorithm for event boundary detection is based on an outlying sensor detection algorithm. For an event sensor, there often exist two regions, with each containing the sensor, such that the absolute value of the difference between the reading of the sensor and the median reading from all other sensors in one region is much larger than that in another region. These approaches are not efficient because they do not consider the temporal correlation of sensor readings.

Bettencourt et al. [50] proposed an outlier detection technique to identify anomalous events and errors in ecological applications of distributed sensor networks. This method uses spatio-temporal correlation of sensor data to distinguish erroneous measurements and events. A measurement is considered to be an outlier when its value in the statistical significance test is less than the user-specified threshold. The drawback of this approach is dependence on the user-specified threshold.

Jun et al. [51] presents a statistically based approach, which uses alpha-stable distribution. The proposed algorithm consists of collaborative time-series estimation, variogram application, and principle component analysis (PCA). Each node detects any temporally abnormal data and transmits the verified data to a local cluster-head, which detects any survived spatial outlier and determines the faulty sensors accordingly. Their approach achieves 94% accuracy when the noise level is alpha = 0.9 although alpha-stable distribution might be considered for real sensor data and a cluster-based structure may be susceptible to dynamic changes of network topology.

1.6.2 Nonparametric Approaches

These approaches have no knowledge about the underlying data distribution. They typically define a distance measure to identify outliers. Outliers are those points that are distant from their own neighborhood in a data set. Various detection techniques are available with a wide range of parameters. Parametric methods are not flexible enough like nonparametric methods, but due to dimensionality and computational complexity, the efficiency might deteriorate in some cases. Two widely used approaches in this category are discussed as follows:

1.6.2.1 Histogramming

It is a model that involves counting the frequency of occurrence of different data instances and then compares the test instance with each of the histogram categories to test whether it belongs to any of them.

Sheng et al. [52] proposed a histogram-based technique for outlier detection to reduce the communication cost for data collection applications of sensor networks. Rather than collecting all the data in one location for centralized processing, they propose collecting hints about the data distribution and using the hints to filter out unnecessary data and identify potential outliers. The main drawbacks of this technique are communication overhead and one-dimensional data.

1.6.2.2 Kernel Function

This function is used to estimate the probability distribution function (pdf) of the normal instances. Data instances that lie in the low probability area of the pdf are declared as outliers.

Palapans et al. [53] proposed a technique for online deviation detection in streaming data. They discussed how their technique can be operated efficiently in the distributed environment of a sensor network. In the sensor data, a value is considered as an outlier if the number of values in its neighborhood is less than a user-specified threshold. This technique can also be implemented for identification of an outlier in a more global perspective. The main problem of this technique is the user-defined threshold.

1.6.3 Strength and Weakness

Statistical approaches are holding a strong mathematical background to detect outliers. But parametric approaches are not feasible when the prior knowledge of the data distribution is not available and hence quite useless in many aspects. But nonparametric methods are quite useful compared to parametric methods because the data distribution knowledge is not required. However, these methods might have high computational complexity due to high-dimensional data sets. Also user-defined parameters are not easy to set.

1.7 CONCLUSION

Outlier detection is an interesting arena of computer and network security. It is also applied in various application domains. It is regarded as one of the fundamental problems of data mining as well. In this chapter, we have summarized outlier detection techniques along with various research direction and application domains. We have described existing approaches for outlier detection techniques and believe that our contribution will help the reader to easily understand the particular research focus in this chapter.

AUTHORS' BIOGRAPHIES

Mohiuddin Ahmed is working in the arena of data mining and network security toward his PhD degree at the University of New South Wales, Canberra. He received his bachelor of science degree in computer science and information technology from Islamic University of Technology, Bangladesh, in 2011.

Abdun Naser Mahmood received the BSc degree in applied physics and electronics and the MSc degree in computer science from the University of Dhaka, Bangladesh, in 1997 and 1999, respectively. He completed his PhD degree from the University of Melbourne in 2008. He joined the University of Dhaka as a lecturer in 2000 and as assistant professor in 2003, when he took a leave of absence for his PhD studies. Currently, he is working as a lecturer at the University of New South Wales, Canberra, with the School of Engineering and Information Technology. His research interests include data mining techniques for network monitoring and algorithm design for adaptive sorting and sampling.

Jiankun Hu obtained his masters degree from the Department of Computer Science and Software Engineering at Monash University, Australia and his PhD degree in control engineering at Harbin Institute of Technology, China. He has been awarded the German Alexander von Humboldt Fellowship working at Ruhr University, Germany. He is currently a professor of cyber security at the School of Engineering and Information Technology, UNSW Canberra. Dr. Hu's current research interests are in network security with an emphasis on biometric security, mobile template protection, and anomaly intrusion detection. These research activities have been funded by three Australia Research Council (ARC) Grants. His research work has been published in top international journals.

REFERENCES

1. V. J. Hodge and J. Austin (2003). A survey of outlier detection methodologies. *Artificial Intelligence Review*, vol. 22, pp. 85–126.
2. D. Hawkins (1980). *Identification of Outliers*. London: Chapman and Hall.
3. V. Barnett and T. Lewis (1994). *Outliers in Statistical Data*. New York: John Wiley Sons.
4. D. Yu, G. Sheikholeslami, and A. Zhang (2002). Findout: Finding outliers in very large datasets. *Journal of Knowledge and Information Systems*, vol. 4, no. 3, pp. 387–412.
5. M. F. Jiang, S. S. Tseng, and C. M. Su (2001). Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, vol. 22, no. 6–7, pp. 691–700.
6. P. J. Rousseeuw and A. M. Leroy (1996). *Robust Regression and Outlier Detection*. John Wiley and Sons.
7. J. Laurikkala, M. Juhola, and E. Kentala (2000). Informal identification of outliers in medical data. In *Proceedings of IDAMAP*.
8. M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander (2000). LOF: Identifying density-based local outliers. In *Proceedings of ACM SIGMOD*, pp. 93–104.
9. S. Harkins, H. He, G. J. Willams, and R. A. Baster (2002). Outlier detection using replicator neural networks. In *Proceedings of DaWaK*, pp. 170–180.
10. B. Schlkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson (2001). Estimating the support of a high dimensional distribution. *Neural Computation*, vol. 13, no. 7, pp. 1443–1471.
11. S. Shekhar, C.-T. Lu, and P. Zhang (2001). A unified approach to spatial outliers detection. *GeoInformatica*, vol. 7, no. 2, pp. 139–166.
12. A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley (2000). Physio Bank, Physio Toolkit, and Physio Net: Components of a new research resource for complex physiologic signals. *Circulation*, vol. 101, no. 23, pp. 215–220, http://circ.ahajournals.org/cgi/content/full/101/23/e215.
13. A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar (2003). A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of SIAM*.
14. R. Fujimaki, T. Yairi, and K. Machida (2005). An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. New York: ACM Press, pp. 401–410.
15. D. J. Marchette (2001). *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*. New York: Springer.

16. S. E. Smaha (1988). Haystack: An intrusion detection system. In *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, USA, December 1988. Los Alamitos, CA: IEEE Computer Society Press.
17. R. J. Bolton, and D. J. Hand (2001). Unsupervised profiling methods for fraud detection. In *Proceedings of CSCC*.
18. Z. He, X. Xu, and S. Deng (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, vol. 24, no. 9–10, p. 1641.
19. P. L. Brockett, X. Xia, and R. A. Derrig (1998). Using Kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. *Journal of Risk and Insurance*, vol. 65, no. 2 (June), pp. 245–274.
20. A. Arning, R. Agrawal, and P. Raghavan (1996). A linear method for deviation detection in large databases. In *Proceedings of 2nd International Conference of Knowledge Discovery and Data Mining*, pp. 164–169.
21. J. Lin, A. E. Fu, and H. V. Herle (2005). Approximations to magic: Finding unusual medical time series. In *Proceedings of Symposium on Computer-Based Medical Systems*, Washington, DC, USA, pp. 329–334.
22. V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris (2006). Hierarchical anomaly detection in distributed large-scale sensor networks. In *ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications*. Washington, DC: IEEE Computer Society, pp. 761–767.
23. M. Augusteijn and B. Folkert (2002). Neural network classification and novelty detection. *International Journal on Remote Sensing*, vol. 23, no. 14, pp. 2891–2902.
24. Y. L. Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson (1990). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, vol. 396, p. 404.
25. C. Spence, L. Parra, and P. Sajda (2001). Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*. Washington, DC: IEEE Computer Society, p. 3.
26. H. Dutta, C. Giannella, K. Borne, and H. Kargupta (2007). Distributed top-k outlier detection in astronomy catalogs using the demac system. In *Proceedings of 7th SIAM International Conference on Data Mining*.
27. K. Kadota, D. Tominaga, Y. Akiyama, and K. Takahashi (2003). Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample classification. *Chem-Bio Informatics*, vol. 3, no. 1, pp. 30–45.
28. S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, and P. Tavan (2000). Generalized radial basis function networks for classification and novelty detection: Self-organization of optional Bayesian decision. *Neural Networks*, vol. 13, no. 10, pp. 1075–1093.
29. P. Crook and G. Hayes (2001). A robot implementation of a biologically inspired method for novelty detection. In *Proceedings of Towards Intelligent Mobile Robots Conference*. Manchester, UK.
30. E. M. Knorr and R. T. Ng (1998). Algorithms for mining distance-based outliers in large datasets. *Proceedings of the VLDB Conference*, New York, USA, pp. 392–403.
31. Y. Yang, T. Pierce, and J. Carbonell (1998). A study on retrospective and on-line event detection, *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp. 28–36.
32. G. S. David Sam Jayakumar and B. John Thomas (2013). A new procedure of clustering based on multivariate outlier detection. *Journal of Data Science*, vol. 11, pp. 69–84.
33. H. Hotelling (1951). A generalized T test and measure of multivariate dispersion. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (Edited by J. Neyman), Berkeley, CA: University of California Press, pp. 23–41.
34. M. Belal Al-Zoubi, A. Al-Dahoud Ali, and A. Abdelfatah Yahya (2010). Fuzzy clustering-based approach for outlier detection, *Proceedings of the 9th WSEAS International Conference on Applications of Computer Engineering*.
35. D. Pham (2001). Spatial models for fuzzy clustering, *Computer Vision and Image Understanding*, vol. 84, no. 2, pp. 285–297.
36. C. L. Blake and C. J. Merz (1998). UCI repository of machine learning databases, http://www.ics.uci.edu/mlearn/MLRepository.html, Department of Information and Computer Sciences. Irvine, CA: University of California.
37. K. Yoon, O. Kwon, and D. Bae (2007). An approach to outlier detection of software measurement data using the K-means clustering method, *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, Madrid, pp. 443–445.

38. A. Loureiro, L. Torgo, and C. Soares (2004). Outlier detection using clustering methods: A data cleaning application. In *Proceedings of KDNet Symposium on Knowledge-Based Systems for the Public Sector*. Bonn, Germany.
39. M. Belal Al-Zoubi, A. Al-Dahoud, and A. Abdelfatah Yahya (2010). New outlier detection method based on fuzzy clustering. *WSEAS Transaction on Information Science and Application*, vol. 7, no. 5, May.
40. V. Hautamäki, S. Cherednichenko, I. Kärkkäinen, T. Kinnunen, and P. Fränti (2005). Improving k-means by outlier removal, in *Proc. 14th Scandinavian Conference on Image Analysis (SCIA '05)*, pp. 978–987.
41. D. Ren, I. Rahal, and W. Perrizo (2004). A vertical outlier detection algorithm with clusters as by-product. In *Proceedings of ICTAI*.
42. H. Fan, O. Zaïane, A. Foss, and J. Wu (2006). Nonparametric outlier detection for efficiently discovering top-*n* outliers from engineering data. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, Singapore.
43. W. Jin, A. Tung, and J. Han (2001). Mining top-*n* local outliers in large databases. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, San Francisco, CA.
44. S. Ramaswamy, R. Rastogi, and K. Shim (2000). Efficient algorithms for mining outliers from large data sets. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD)*, Dallas, TX.
45. H. Zengyou, D. Shengchun, and X. Xiaofei (2002). Outlier detection integrating semantic knowledge. *Advances in Web-Age Information, Management*, Lecture Notes in Computer Science, vol. 2419, pp. 126–131.
46. S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos (2003). LOCI: Fast outlier detection using the local correlation integral. In *Proc. ICDE*.
47. K. Zhang, M. Hutter, and H. Jin (2009). A new local distance based outlier detection approach for scattered real world data. In *Proc. PAKDD*.
48. H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek (2009). Loop: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09*, New York, USA, pp. 1649–1652.
49. W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng (2007). Localized outlying and boundary data detection in sensor networks, *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1145–1157.
50. L. A. Bettencourt, A. Hagberg, and L. Larkey (2007). Separating the wheat from the chaff: Practical anomaly detection schemes in ecological applications of distributed sensor networks, *Proc. IEEE International Conference on Distributed Computing in Sensor Systems*.
51. M. C. Jun, H. Jeong, and C. C. J. Kuo (2006). Distributed spatio-temporal outlier detection in sensor networks, *Proc. SPIE*.
52. B. Sheng, Q. Li, W. Mao, and W. Jin (2007). Outlier detection in sensor networks, Proc. MobiHoc.
53. T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos (2003). Distributed deviation detection in sensor networks, *ACM Special Interest Group on Management of Data*, pp. 77–82.

2 Network Traffic Monitoring and Analysis

Jeferson Wilian de Godoy Stênico and Lee Luan Ling

CONTENTS

2.1 INTRODUCTION

At their very first stage, computer networks were invented in order to be able to share documents and devices. With significant advances in electronics and signal processing technologies during past decades, accessing, searching, and sharing information via communication networks (e.g., the Internet) have become a part of our everyday life. Because frequently a great part of this information is confidential or has restricted access only for authorized persons, preventing unauthorized use and accessing this information and detecting intruders have become an indispensable mechanism for all information systems and communication networks. Efficient prevention mechanisms require a combination of the expertise of security technicians and powerful hardware and software tools in order to be able to achieve high levels of security management, especially for today's modern computing systems. An intrusion detection system (IDS), by its turn, is one of these security tools for this end.

An intrusion detection system is used to monitor the security status of networks by detection of external invasions and abnormal operations of servers, whether contaminated or not. The information provided by an IDS is crucial for the guarantee and protection of integrity, privacy, and authenticity of the flowing data in networks. Another major function of an IDS is attack prevention by learning. From a real attacking experience, the data should be carefully analyzed, the origin of the attack should be determined, and the level of hazards and malicious penetration should be measured in order to improve the security system's ability to prevent any future attacks.

2.1.1 Evolution of Intrusion Detection Systems

A very first research work on intrusion detection (ID), titled *Computer Security Threat Monitoring and Surveillance* [1], appeared in 1980, in which the concept of an *audit trail* was presented. *Audit trails* provide valuable information that can be used for tracking misuses of information systems and understanding user's behavior. This research work had established the foundation for the later design and development of intrusion detection systems (IDS).

Dorothy E. Denning, of SRI International,* in 1983, participated in a project that resulted in significant advances in IDS development. She examined the records of government mainframes and created users' profiles according to their activities.

In 1984, Denning collaborated with the construction of the first intrusion detection system, the *Intrusion Detection Expert System* (IDES) [2]. IDES can be viewed as the first functional IDS. During the same year, SRI International presented a method to analyze information used for user authentication on the records of ARPANET (the precursor of Internet).

Later, in 1987, based on her wide experience in IDS research and development, Denning published an article called "An Intrusion Detection Model" [3], which had served as an important guidance for new IDS development. In spite of all these research efforts, the real and significant advances in intrusion detection were demonstrated by the Haystack project in 1988 [4]. This project improved the conventional IDS by incorporating a new approach, resulting in the so-called distributed intrusion detection system (DIDS) [5]. The DIDS introduced techniques for data pattern detection, which consisted of a comparison of the input data with predefined standard data patterns. Later, in 1989, a trade company named Haystack Labs, composed of some researchers of the Haystack project, released the latest generation of the IDS technology, called Stalker. Stalker was a host-based intrusion detection system (HIDS) having robust capabilities of information searching on record systems (manually or automatically via the methodology of questions). In summary, the Stalker technology and the methodologies developed by SRI International and Dorothy E. Denning constituted the major advances in the technology of host-based IDS (HIDSs) in the 1980s.

In the early 1990s, Heberlein et al. suggested the concept of network-based intrusion detection systems (NIDS) [6]. In fact, they were the major inventors of network security monitor (NSM), the

* http://www.sri.com/

first NIDS implemented and used by important government organizations. NSM was designed to extract information through network traffic analysis. NSM had motivated significant increasing interest in intrusion detection inside the research community as well as in the investments in NIDS production. Later, the group of Heberlein and some staff of Haystack jointly went beyond their initial objectives of the NIDS project, extending to and producing the hybrid intrusion detection systems (HIDSs/NIDSs).

Still in early 1990s, several companies started to release their commercialized IDS versions. Among them, the Haystack Labs was the first company marketing its IDS tools, more specifically, the host-market based Stalker. The Science Applications International Corporation (SAIC)* had developed a host-based detection system called the computer misuse detection system (CMDS) [7]. Simultaneously, the US Air Force's Cryptologic Support Center developed the automated security measurement system (ASIM) to monitor network traffic on the US Air Force's communication network [8]. ASIM was probably the very first IDS that had achieved considerable breakthrough in terms of scalability and portability, the main issues that had plagued NID products since the beginning. Moreover, ASIM was considered as the first IDS capable of incorporating both hardware and software solutions into network intrusion detection systems.

However, the commercialization of IDS products became significant in the market only in the middle of 1997. During that year, Internet Security Systems (ISS) was the market leader among most commercially available products because of its outstanding and distinguished NIDS solution, called RealSecure. A year later, Cisco acquired Wheel Group in order to be able to compete with ISS. Centrax Corporation, a new company formed by some previous Haystack Labs' researchers and SAIC's CMDS team had also succeeded with its HIDS solution. From then on, many new companies marketing IDSs have emerged by corporation merging and acquisition.

According to reported statistics [9], currently the IDSs are one of the most used security devices. In fact, IDS technologies occupy the seventh position in terms of number of usages for network security purposes (see Figure 2.1).

Moreover, the employment of IDS technology has remarkably increased in comparison with other security technology types. As illustrated by Figure 2.1, there was almost a 25% increase in IDS usage from 1998 to 2002.

This chapter aims to provide a comprehensible description of intrusion detection systems and their evolution, implementation, and operation inside communication network environments. Precisely, the description will be carried out in the context of network monitoring and network traffic analysis. For easy understanding of all addressed issues, we make use of currently widely employed security tools as examples to illustrate the details of host-based intrusion detection systems (HIDSs) and network-based intrusion detection system (NIDSs) (Figure 2.2).

2.1.2 Classification of Intrusion Detection Systems

As mentioned previously, intrusion detection systems aim to provide plausible traces of information systems being invaded. Figure 2.3 shows a complete characterization and classification of intrusion detection systems.

2.1.2.1 Intrusion Detection Categories

2.1.2.1.1 Misuse Intrusion Detection Systems

Misuse intrusion detection systems are designed to search for already known patterns of attacks and intrusions. The systems are also named as signature-based intrusion detection systems or knowledge-based intrusion detection systems. Some interesting features of these kinds of intrusion detection systems are that (a) the characteristics of the attacks are well known and (b) the attack patterns can

* www.saic.com

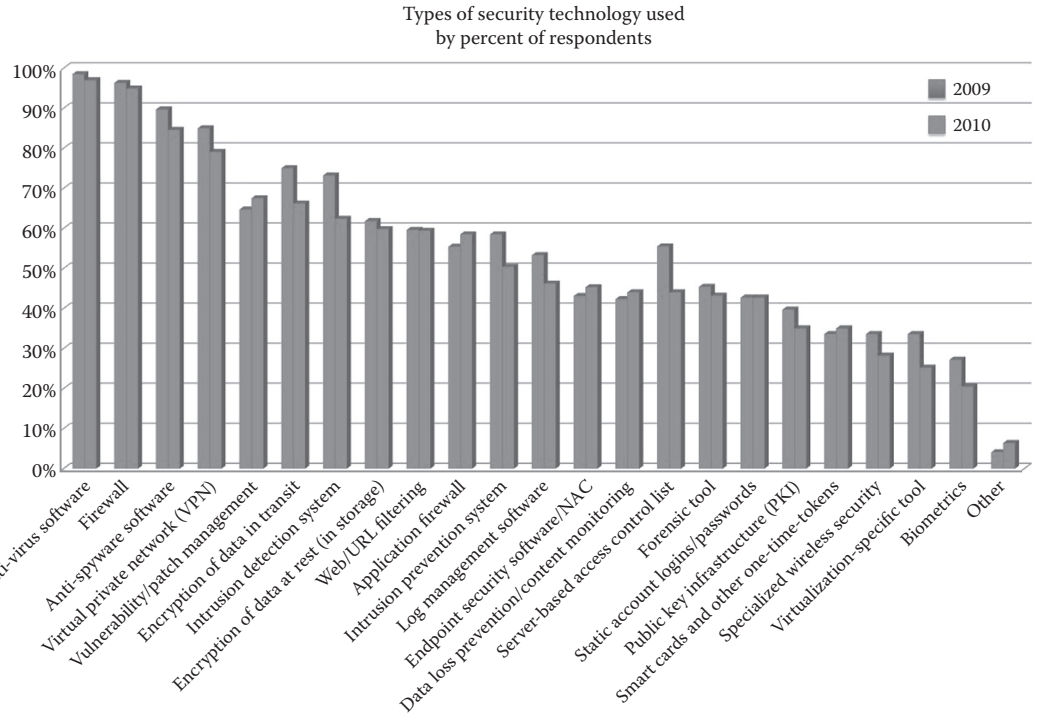

FIGURE 2.1 Statistics of usages of security technologies. (From 15th Annual Computer Crime and Security Survey, CSI/FBI, 2010/2011: https://cours.etsmtl.ca/log619/documents/divers/CSIsurvey2010.pdf.)

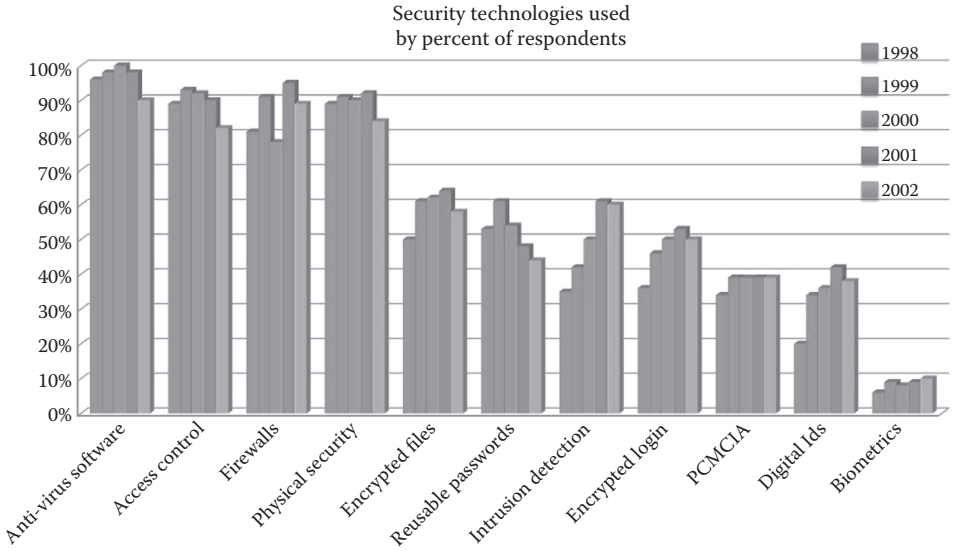

FIGURE 2.2 Evolution of usages of security technologies from 1998 to 2002. (From Computer Crime and Security Survey, CSI/FBI, VIII, 1, Spring 2002. http://diogenesllc.com/2002cybercrimesurvey.pdf.)

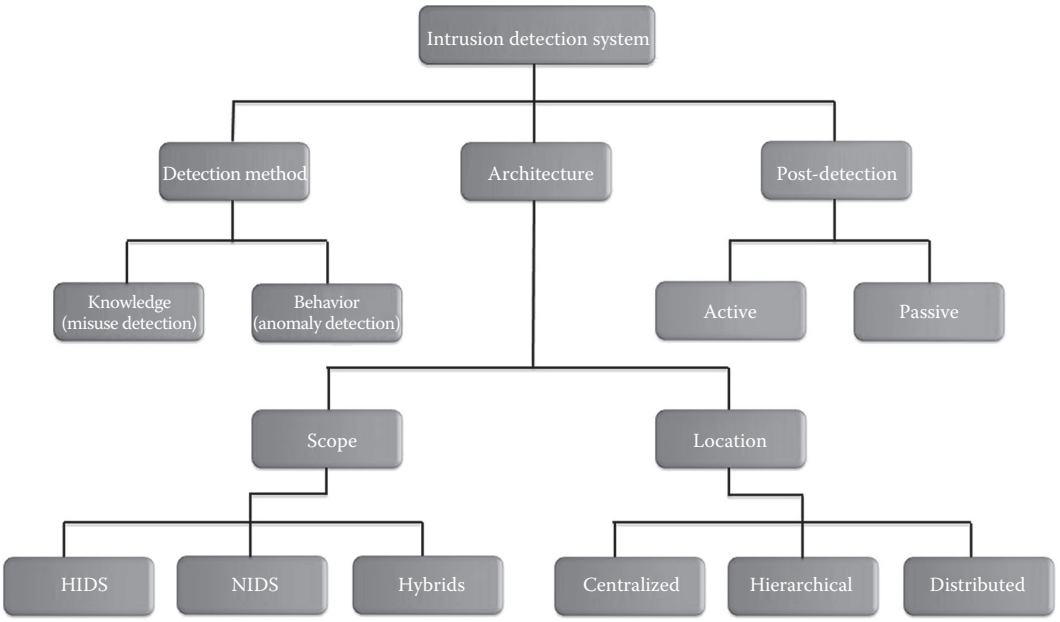

FIGURE 2.3 Characterization and classification of intrusion detection systems.

be easily encoded into an expert system. In other words, this detection system category usually holds a database of known attack signatures that need to be updated whenever new attack types are identified.

A well-known encoded instruction (or rule) in a misuse IDS for detecting a denial of service attack is called the "Ping of Death," which is one of the oldest network attack types [11]. This kind of attack is mainly characterized by sending a very large ICMP echo request packet, intending to provoke an unavailability of network services among connected multiple operating systems by generating bugs or causing programming errors in vulnerable software. An "ICMP echo request" packet thus is considered as an attack when its size is superior to 65.535 bytes. Notice that this encoded rule simultaneously belongs to misuse-based and network-based IDSs.

When an encoded instruction focuses on events related to any specific isolated information system or computer, it is specified as an instruction of the host-based intrusion detection system (HIDS). Examples of this kind are (a) alerts caused by excessive processor occupancy time when a particular sequence of calls to operating system functions occurs and (b) certain critical files are changed.

IDSs also detect intrusions based on knowledge. In other words, intrusions are interpreted in terms of knowledge, which defines system states according to actions performed by the system. Pairs of attributes-values are used to represent the possible system. The transition of state is provoked by some actions. Figure 2.4 illustrates the identification of intrusion by the knowledge-based IDS.

Independent of the focus of an IDS (network or host based), a misuse IDS requires the incorporation of human expert knowledge into the IDS in order to be able to operate adequately. A main disadvantage of a misuse intrusion detection system is its inability to detect new attack types because novel attack information has not been added to the standard database yet. As a result, this

FIGURE 2.4 Sequence of changing states in knowledge-based ID systems.

fact has motivated the flourish of research and development of anomaly-based intrusion detection approaches.

2.1.2.1.2 Anomaly-Based Intrusion Detection

Security systems for anomaly intrusion detection, also called behavior-based intrusion detection systems, seek to determine or create models that represent the normal or expected behavior of computational systems or communication networks. An alert will be raised whenever deviations from the expected behavior are found. It is presumed that involved system intrusion or attack activities are parts of the subset composed of anomalous activities. Ideally, it is desirable that the set of all malicious activities is exactly equal to the set of all anomalous activities. Under this idealized situation, the system will generate zero false-positive and false-negative errors. However, in practice, all four possible situations occur with nonzero probabilities:

- *Intrusive activity but normally declared:* This situation, also known as false-negative, is extremely dangerous because the system fails to detect an attack or intrusion.
- *Nonintrusive and normal activity:* Also called the positive-negative condition. Because it consists of a normal and nonintrusive activity, the monitoring system will not be activated.
- *Nonintrusive and abnormal activity:* Also called the false-positive condition. The system alerts, indicating the detection of an intrusion which has not actually occurred.
- *Intrusive and abnormal activity:* An ideal situation in which the system correctly detects an attack or intrusion.

Figure 2.5 illustrates all four possible situations through one-dimensional probability distributions (probability distributions of normal and abnormal activities).

2.1.2.2 Intrusion Detection System Architecture

The architecture of intrusion detection systems should be elaborated, taking into account several relevant issues: (a) what source of information is being monitored ("network," "host computer," or "hybrid" entities), (b) how tasks are being distributed, and (c) what kinds of processing modules comprise the IDS. Through a classic example, Figure 2.6 illustrates several aspects commonly encountered in elaborating solutions for intrusion detection in practice. The given example shows a LAN (possibly representing networks of corporations or universities) connected to the Internet by a "firewall." The firewall is a security device that separates different networks; however, all are connected to a distinct network that provides Internet service and forms a so-called

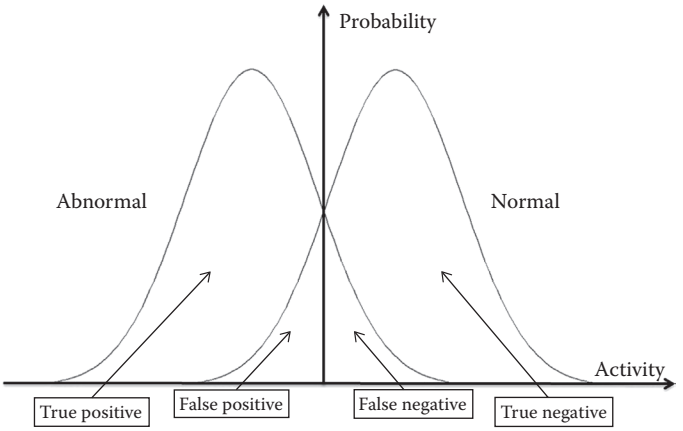

FIGURE 2.5 Four situations defined by probability distributions of normal and abnormal activities.

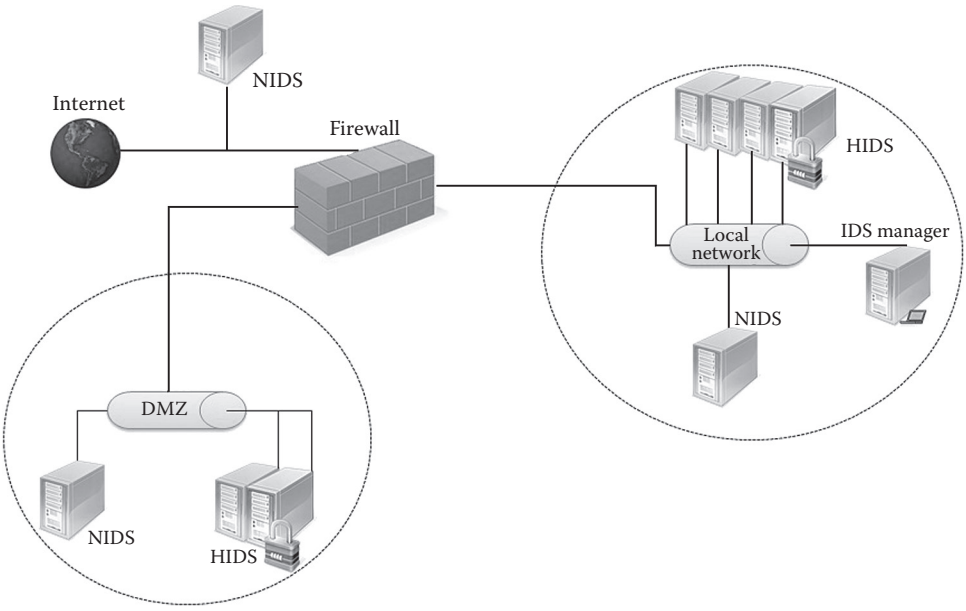

FIGURE 2.6 Example of classic intrusion detection solution.

DMZ ("demilitarized zone"). Each segment of this connected network, (local network, DMZ, and Internet) holds a network-based intrusion detection system (NIDS) monitoring the corresponding network segment. In addition, those servers that are considered critical may have a host-based intrusion detection system (HIDS) implemented and used to monitor some specific events possibly happening on these servers. Finally, there is a management tool of IDS at LAN used to collect and organize the information received from various connected modules.

2.1.2.2.1 Scopes of IDSs

In terms of the analyzed and monitored source of information, an IDS can be classified into the group of host-based intrusion detection systems, network-based intrusion detection systems, or hybrid intrusion detection systems. Each kind of IDS has its peculiar approach for problem formulation and solution elaboration. In this section, we describe the problems and solutions encountered as well as the advantages and disadvantages of each IDS category.

2.1.2.2.1.1 Host-Based Intrusion Detection Systems (HIDS) A host-based intrusion detection system is used to monitor the behavior or dynamic states of a machine. To this end, it consults its log analysis files or data from the audit agents and thus detects and assesses changes in the file system, user access control, behavior of system processes, and use of resources among others. HIDS can also perform simple and concise inspections of packets arriving from the network on which the machine is connected. A typical example that involves HIDSs is record systems. Figure 2.7 shows the architecture of a recording system, in which a HIDS uses recorded information as a standard pattern.

Another feature found in many HIDS is the hash of the file system. HIDS creates an initial database with hash values of files that are considered important. To identify any violation of recorded data, the hash value is recalculated and compared with the correct one previously saved in the database. If any discrepancy between two hash values appears, it implies some changes have occurred. This hash number verification mechanism is very effective for the following actions: detection of rootkit installation,* inclusion of new users, or setting changes. Once any suspicious activity is

* A rootkit is a specific type of malware that runs actions to intercept the operational system and change the results.

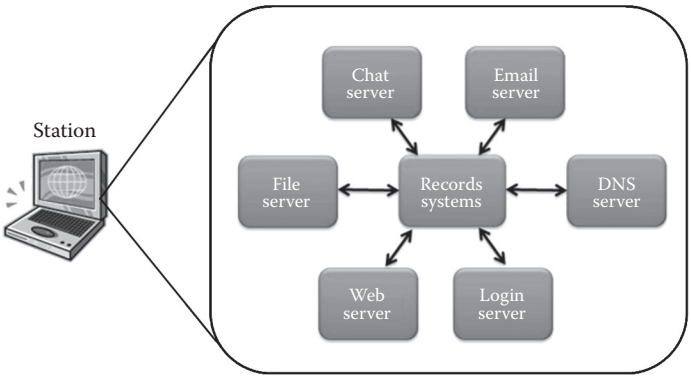

FIGURE 2.7 Architecture of a recording system.

detected after performing hash data comparison and analysis, the HIDS can either generate an alert or place in the logs the encountered suspicious activity.

Among many benefits from using a HIDS, we can highlight the following [12]:

- Establishing a strong association between programs and users: identifying the author and the time at which applications or commands were executed.
- Because of being one of the potential targets of attack, HIDS is capable of collecting considerable and valuable information during the attack.
- Confronting coming threats directed only to the machine where the HIDS resides without the necessity of capturing whole network traffic.
- Direct access to the system with priority and without any dependence on other means to access data.
- No need for additional hardware.

Among the disadvantages of using a HIDS, we can cite the following [13]:

- High complexity in management and scalability of HIDS (because of the necessity of configuration and constant observation on all hosts, impacting scalability).
- Operational system dependent.
- Susceptible to attacks, resulting in corrupted data or data deletion. This is due to the fact of its priority in direct access and belonging to or being part of the host environment.
- Possibly becoming correlated with attack data and requiring some additional performance data correlation analysis on data from all involved network hosts.
- Deep data analysis and very detailed information collection can affect negatively the performance of host monitoring.

2.1.2.2.1.2 Network-Based Intrusion Detection Systems (NIDS) The NIDS can monitor data collected from its own network segment or from multiple network hosts. Currently, there are many commercially available NIDS tools, for example, NADIR [14] and DIDS [15]. Many of these tools have implemented data collecting facilities. Although the architecture of NIDS may vary among these tools, according to Hofmeyr et al. [16], basically, it is composed of three units: a data collector, a manager, and a communication module responsible for transmission and receiving of data and analysis results.

The collector usually has a set of sensors, possibly distributed over the network, and is responsible for the data capture and formatting as well as network traffic analysis. The collector can be viewed as a NIDS component responsible for data preprocessing and preliminary network traffic

analysis. On the collector, the collected input information is processed and normalized so that it becomes prepared and eventually used to compare with the standard anomaly profiles. After having entry data standardized, the classification mechanism implemented on the manager will attribute (classify) the input information into one of two possible events with intrusive behavior or normally behaved. Notice that the event classification mechanism has distinguished implementation according to IDS types, i.e., misuse-based detection and anomaly-based detection. The manager itself is also responsible for two important tasks: the integrated management of sensors and the specification of response patterns for each network behavior type. Finally, the most basic module—the communicator—executes the tasks of transmitting and receiving data and analysis results according to the policies set by administrators.

For anomaly detection, the IDS captures packets, analyzes their headers, and compares them with known patterns or signatures. The NIDS found efficiency against several different types of attack and can block some of them in real time.

For misuse-based detection, the classification mechanism compares the input data with instructions and other behavior descriptors. On the other hand, for anomaly-based detection, the comparison is usually based on the statistical profiles of the user's or the system's historical behaviors. For this end, a default behavior of each user is outlined and used as the standard pattern for comparison with captured data [17].

Among some remarking benefits of using the NIDS we point out the following [12,13]:

- No impact on network performance.
- Attacks can be identified in real time, thus enabling the administrator to respond to the attacks quickly.
- Able to capture suspicious attacks that have not yet been identified.
- Operational system independent.
- Possibly being invisible to attackers. NIDS does not leave any trace of its network monitoring actions whereas a HIDS certainly leaves its proper vestiges in the system where it is installed.
- Having wide areas of applications and requiring simple and concise infrastructures.

Among some negative views of the NIDS, we outline the following [12,13]:

- May require highly complex protocols for specific applications
- Inability to monitor encrypted traffic
- Restricted applications on fragmented networks, especially those with switches
- Inability to perform traffic analysis when the volume of network traffic traces exceeds the system's collecting capacity
- Need of large recording capacity for data storage (for instance, monitoring the states of TCP connections with large network flows)

2.1.2.2.1.3 Hybrids The conception of the hybrid IDSs were based on the possibility of maximizing the strengths of both HIDS and NIDS. In practice, a hybrid IDS operates as a NIDS, collecting and processing network traffic in order to detect attacks. On the other hand, as a HIDS, the hybrid system focuses on each host, only processing the packets addressed to its own system. The hybrid system can resolve the low performance problem of NIDS but still leave the scalability as an open problem (IDS is required to be installed on equipment).

One author [18] argues that both NIDS and HIDS can be characterized by two distinct methods (knowledge-based detection or misuse detection and behavior-based detection or anomaly detection), and a hybrid IDS is viewed as an IDS that combines these two detection methods. In general the pattern-based strategy provides a low false alarm rate but fails to identify unpublished attacks. On the other hand, a behavior-based approach can provoke a large number of alerts, and not all of

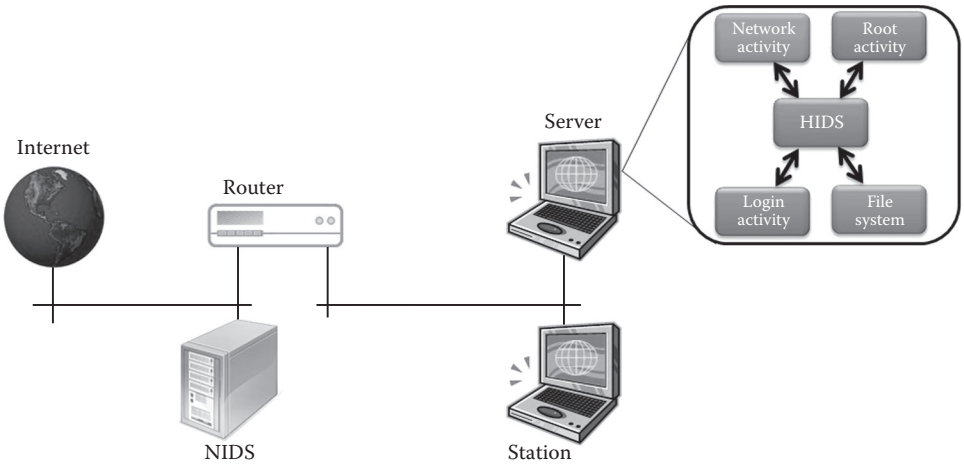

FIGURE 2.8 An example configuration of hybrid intrusion detection systems.

them are indeed attack attempts. An interesting aspect of behavior-based approach is its ability to detect new attack types.

Another important feature of a hybrid IDS is its centralized management; i.e., the IDS can locate sensors in various network segments and other host-based IDSs used in machines. The management center can control the rules or instructions for both IDS types, thus forming a hybrid IDS [19]. Figure 2.8 shows a widely employed configuration of a hybrid IDS in which a NIDS belongs to a local network and HIDS run on their corresponding master servers.

2.1.2.2.2 Location and Distribution of IDSs

Practical implementation of an intrusion detection system means a functional and operational combination of monitoring agents (network sensors or hosts equipped with data processing modules), detection modules, and management tools. Figure 2.9 shows how these processing and operational modules are organized and functionally related in the IDS. According to the network architecture

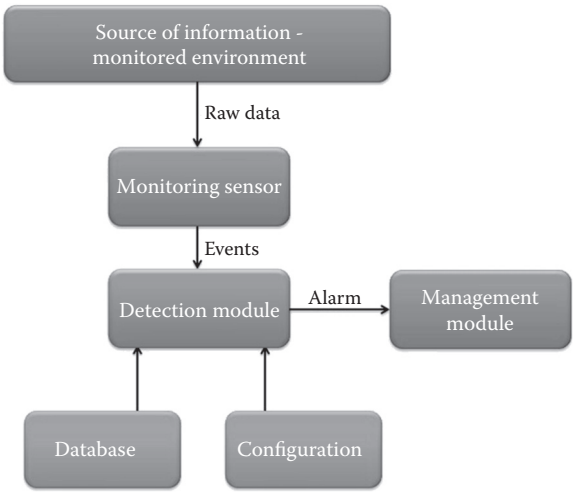

FIGURE 2.9 Elements and a functional diagram of intrusion detection systems.

and location of these modules, an IDS can be classified into one of the following categories: centralized, hierarchical, and distributed.

2.1.2.2.2.1 Centralized IDS A centralized IDS, in principle, has only one manager responsible for the event analysis, detection, classification, and system action. One or more monitoring modules (also called agents) can be employed, responsible for data collecting and transmission to the central module (manager). Under this IDS architecture model, the central module also is known as the agent-manager.

2.1.2.2.2.2 Hierarchical IDS Both hierarchical and centralized ID systems belong to the agent-manager model category. However, the former ones allow some variation of system configuration on several different subordination levels. In a centralized IDS, all monitoring agents are subordinated (sending information and reporting results) to the same central module or manager. The hierarchical system model permits distinct subordinations. Agents that collect information regarding the network (NIDS) can be subordinated to the network manager module. Agents that monitor specific computers (HIDS) can be subordinated to the host manager module.

2.1.2.2.2.3 Distributed IDS A distributed intrusion detection system permits data processing and analysis and event detection being partially or fully executed by the collecting modules (agents). This task distribution strategy implies that the system intelligence is distributed among the manager modules and agents that make up the ID system.

2.1.2.3 Actions of Post-Detection

Once having detected an intrusion, the IDS initiates certain actions of post-detection in responding to the detected intrusions. The actions of post-detection can be classified into two classes: active and passive actions.

2.1.2.3.1 Active Actions

With the declaration of detected intrusion, the IDS may react actively to protect the threatened entity against the attack. This kind of ID system class has been the subject of recent research works and, due to its importance, has earned a specific name: intrusion prevention systems (IPS). For example, during a TCP connection, an active and network-based IDS may send some specific purposed packets (TCP packets with FIN and RST flags) to the source entity of traffic flows to interrupt the connection (which is considered as an intrusion). In addition, the IPS can interact with edge routers (routers located at the perimeter between the local networks of an organization and external networks to the Internet) or firewalls and establish rules to filter certain packet flows originating from some specific source addresses.

2.1.2.3.2 Passive Actions

A passive classified IDS only generates and sends notifications to the system operator or administrator through the expert management console, after having detected attack patterns. These notifications or alerts can be generated and transmitted via various standard protocols, e.g., email, SNMP alerts, etc. The intrusion detection exchange format [21], a working group belonging to the IETF (Internet Engineering Task Force) [20], was created to standardize the communication mechanisms as well as the communication language between ID systems, also between agents and detection management modules. The RFC (request for comments) documents, released by this working group for the standardization of exchanged information between modules of IDS or between several different IDSs, deal with the following issues:

- Intrusion detection message exchange requirements
- Intrusion detection message exchange format
- Intrusion detection exchange protocol (IDXP)

2.2 NETWORK MONITORING

Network monitoring is an extremely important task in combating system intrusion. The monitoring procedure consists of collecting and recording data from the affected network elements in operation. The tasks performed by these network elements vary greatly in their nature, including statistical information collecting, traffic analysis, and problem diagnosis, among others.

Network monitoring can be performed at defined instances or continuously depending on preestablished objectives and tasks. That is, expected information and results require adequate data collecting methods. For example, if one wishes to obtain the information about monthly network availability, it is necessary to monitor the network continuously for at least a month and to register the periods in which the networks were out of function. Another typical example is monitoring networks to extract some information about the periods of peak usage. The procedure for this end may consist of recording traffic continuously for long periods (several days) and counting and comparing amounts of bytes of traffic flows for each short time segment that composes the whole periods. The tasks of network traffic monitoring can be executed using the SNMP (simple network management protocol) [22]. However, this protocol cannot identify which networks or hosts (both internal and external ones) were more demanded.

Two mechanisms can be applied for network monitoring: active and passive. Through an active method, it is necessary to generate testing traffic flows and transmit them in a controlled manner along one or more network paths (routers). During the transmission, the quality of traffic flows as well as the network performances are monitored.

Notice that there are two factors needed to be considered if the active approach is adopted: network performance can be affected by using testing traffic flows and adopting dynamic routing; performance measures can be altered (becoming better or worse) depending on the selected routing paths.

Through the passive method, real traffic flows (or their statistics) are captured at one or more network points and then analyzed [23].

2.2.1 Network Monitoring Systems

A "network monitoring system" is created software to facilitate network monitoring tasks. This computing program tool is an improved programming version of currently existing network analysis algorithms that automate network data collection and analysis.

One of the key features of a monitoring system is the storage of collected data for historical database compilation and system training. The information acquired from past network activities makes network planning (for either short, medium, or long time periods) easier, aiming to ensure acceptable qualities of offered network services. The recorded data have also demonstrated their utility in helping networks operating appropriately according to the preestablished criteria, e.g., SLAs* (service level agreements). It is important that those network service providers monitor networks constantly in order to guarantee the contracted traffic rates and quality of services.

2.2.2 Network Monitoring Technologies

There are many published research works and implemented computational tools available on the subject of traffic monitoring technology. Cottrell [24] gives an extensive list of these computational tools. Computational tools operating on the packet level, such as "tcpdump" [25] and "Wireshark" [26], are quite popular and widely used by network administrators. Although these tools can capture traffic packets easily, they are not allowed to make any arbitrary query over the captured traffic. Furthermore, these tools typically are directed to process traffic flows originating from a single data source.

* An SLA is a part of a service contract in which a service is formally defined.

Authors [27] present a computational tool solution that allows query. The proposed solution consists of distributed collectors capable of receiving data in the NetFlow format and exporting them in the pcap format so that "Wireshark" is able to execute necessary analysis. However, the authors remarked that the amount of information can be huge and recommend that each collector filters (selects) the data of its own interest.

The network monitoring tool called "ntop" [28] is one capable of operating on both packet and flow levels. The capture of traffic information can be done through a plug via Netflow. Although this monitoring tool is capable of release reports in varying formats, it is not allowed to make arbitrary queries elaborated by the network administrator.

Among the monitoring tools operating with Netflow, the "cflowd" [29] was one of the first ones distributed with open source and later gave rise to "flowscan" [30]. Authors [31] claim that monitoring systems operating on the packet level do not scale in high-speed networks. In fact, network-based intrusion detection systems set their focus on packet-level inspection in stub networks. In addition, the authors suggest a framework for real-time backbone monitoring and attack detection based on NetFlow. The given framework operates under centralized administration and requires users to write plugins for each task to be performed.

A traffic monitoring system [32] based on NetFlow has information stored in an Oracle database. The use of the database greatly increases the flexibility in information consultations. However, such an approach still requires storage of information for later processing as well as centralized management.

The authors of the monitoring tool "SMART" [33] argue that the traditional monitoring tools, operating with NetFlow and storing information on disk for later processing, prevent the efficient handling of large traffic volumes. "SMART," in turn, uses flash memories for data storage and processing. According to experimental investigation, "SMART" is capable of processing 30,000 records per second.

The "Gigascope" monitoring system [34,35] can be viewed as a DSMS dedicated for high-speed network monitoring. The tool has been considered the most efficient monitoring system based on its practical results provided by AT&T. Notice that "Gigascope" is a commercial tool under the closed source condition.

Motivated by investigation on the usage of open source DSMS for traffic monitoring, authors [36] made a case study on the "TelegraphCQ" DSMS. Although only a moderate amount of traffic data was analyzed, the study work has provided additional motivation for further investigation and elaboration of open source tools. A good example is "Paquet" [37].

The monitoring tool "Paquet" uses the data stream management system called "Borealis" operating on the packet level and allows that the network administrator makes arbitrary queries on captured traffic by network interfaces or on data files in pcap format.

An alternative to NetFlow is SFlow, which is a traffic monitoring technology defined by RFC 3176 [38]. The RFC document defines a packet sampling mechanism for SFlow agents, a MIB and a data format used for communication between an agent and a SFlow collector. Because it is not required that a SFlow agent keeps any flow information, the implementation of agents is highly simplified. Moreover, the system specifications require precise monitoring of interfaces up to Gigabits speed rates and possibly more. The data collection of the SFlow is based on a statistical packet sampling method. However, as the collector that receives the samples knows the sampling rate, it is possible to accurately reconstruct the sampled traffic [38–40].

2.3 INTRUSION DETECTION ANALYSIS

As mentioned before, there are two categories of network data traffic analysis, namely misuse detection and anomaly detection. The former one consists of a comparative approach searching for one among a group of preestablished attack patterns (signatures), most matching with the coming network traffic flow, which is known as a pattern matching method. For the anomaly detection technique, also known as behavior-based detection, the IDS learns from past network data to establish

normal system behavior patterns (without attacks) and sets boundaries for this pattern. Any deviation beyond these boundaries is treated as an attack.

2.3.1 Pattern Matching Method

It is a methodology used by intrusion detection devices to detect attacks based on signature matching. This method is the simplest and most widely employed approach for attack detection although it has limitations on application generalization. The basic idea of this method is to compare the captured packet with a sequence of data on text or under binary format, i.e., searching for and identifying known attack signatures. Intrusion detection systems employing signature detection methods are required to continuously update the signature database with appeared novelties.

Constant and continuous research work is needed to create new signature patterns in order to maintain IDS in terms of efficiency against any change of and/or new attack schemes and strategies. Notice that a simple change in an attack scheme could be enough to "fool" an IDS, making signatures no longer effective.

Although a task of pattern matching can be quickly executed for attack detection, this may not be fully true with the evolution of networks in terms of size, speed, and complexity, which turns signatures-based IDS very complex. Definitely, a higher processing capacity will be needed to accommodate growing traffic volumes, network expansion, and even sophisticated attack strategies.

The mechanics of a pattern matching technique is of a reactive type, similar to actions carried out against viruses. When criminals create and spread viruses to attack information systems, antivirus companies learn the viruses (attacks) and create the vaccine (antivirus software) to protect information systems.

2.3.2 Protocol Analysis

In general, intrusion detection systems analyze essentially the header contents of protocols (e.g., IP, TCP, UDP, and ICMP). However, a more advanced intrusion detection system can go further, performing the so-called "protocol analysis." The protocol analysis here introduced implies that the ID system has full knowledge about all related communication protocols and is capable of detecting abnormal or suspicious activities on the network through traffic data analysis. In other words, through traffic patterns supported by those protocols, the IDS can detect the violation of the standards established by the protocols. Notice that the protocol analysis facilitates the detection of both known and unknown attack types, based on the principle of protocol violation in terms of traffic abnormalities, rather than focusing on signature pattern verification.

In protocol detection, despite the need to develop specific signatures by protocol, it is classified as a "behavioral" technique, thus any anomaly differentiation in the use of the protocol will be perceived by the sensing device and generate an alert on change.

2.3.3 Statistical and Probabilistic Approach Based ID

The probabilistic approach of ID provides a powerful complement to the signature-based approach. Most developers of intrusion prevention systems rely on Bayesian strategy to implement probabilistic-based detection modules, i.e., learning situations through conditional probability relationships instead of rules or signatures. With this additional knowledge acquired from these probabilistic studies, signature-based IDSs become more sensitive with and oriented to situations, and simultaneously, anomaly-based IDSs are still able to retain their detection capabilities.

The engineering of this probabilistic technique is as follows. First of all, network data are collected and studied. The study work consists of building a Bayesian statistical model using both real attack samples and false positive samples. Notice that a Bayesian system will be robust only if the number of samples is big enough for system training.

The statistical analysis carried out here shows what traffic patterns are most probable to be false positive and those being actually malicious attacks. Then, these learned patterns can be incorporated into and used by IDSs to provide more precise distinction between false positives and malicious traffic and consequently to reduce significantly false positive identification rates. It is important to highlight that IDSs may require a long time period initially for system model learning what may let information systems be vulnerable to attacks during this period.

2.3.4 Neural Network Based ID

Intrusion detection systems based on neural networks, through adaptive learning procedures, have been used to recognize abnormal behaviors found in network traffic. This is an interesting approach due to the fact that neural networks do not take into account user defined parameters.

The neural network must first be trained with clean traffic data, i.e., traffic flows without malicious activity contents. The training should be constantly executed in order to permit that the neural networks learn constant changes of traffic behaviors. Because of their inherited learning ability, neural networks have been considered as the most appropriate tools for abnormality detection. In spite of their plausible quality, neural networks lack the ability to determine the cause of the abnormality. In other words, neural networks can only determine if there is a breach of security, but not its cause. The effort adopted to circumvent this problem is the development of dedicated neural networks that specialize in a specific type of attack. This makes its use extremely efficient for intrusion detection; however, currently, this technology is still restricted to research labs, and no commercial solution has been implemented in corporate environments as yet.

The anomaly detection techniques, despite still being in their early stages, have already become a reality for business solutions. In fact the technologies of intrusion detection still have much to evolve, despite having been investigated and receiving mass investments from the security industry. Currently, publications of new works with a variety of proposals for development of new approaches and algorithms for intrusion detection occur frequently. This fact clearly demonstrates the need for improvement and investment in the both hardware and software technologies of intrusion detection.

Next we present some examples of these propositions:

a) Strict Anomalies Detection

   This detection technique is based on either heuristics or rules (rather than on pattern or signatures matching) and attempts to detect any type of misuse that falls out of normal system operations. This is opposed to signature-based approaches, which can only detect attacks through related signatures previously created.

   One of the most interesting points of this ID technique is that the IDS does not generate false positive (false alarm) events. As a consequence, the employment of such an IDS (based on the model of strict anomaly detection) may be widely accepted due to the fact that this IDS type is more suitable for network environments in which network resources can be, or have already been, well defined.

b) Holistic Analysis

   The holistic approach is seen as the opposite of the conventional view of security [41]. The conventional analysis is also called reductionist analysis.

   Holism is based on the belief that the whole is greater than the sum of its parts, i.e., the whole is indivisible and inseparable. Translating this concept in safety approach means that it is possible to infer the existence of an attack when a group of observations (apparently not correlated) are related to a structure that represents the knowledge of a method that employs an attack at a high level. In other words, the reductionist methods generate an assumption of truth based on observation of a particular action (a bottom-up approach).

   Holistic methods have their reverse engineering, i.e., the method starts from a general knowledge to infer a specific observation (a top-down approach). There are several ways to

implement holistic analysis. One possible implementation consists of a path or track building over a network, in which the main node of the network acts as a final barrier that an attacker needs to overcome in order to reach his final goal.

Another implementation option would be as follows: performing wide and global analyses of possible tactics that an attacker may use and using observed facts to deduce the next tactic that could be used in attack. Note that the holistic model is based on data collected in an environment using reductionist methods; therefore, it is method dependent and needs to be used in conjunction with the reductionist method.

c) Genetic Algorithms

The creation and implementation of genetic algorithms was inspired by the evolutionary theory of Darwin when associated with genetic engineering [42]. Genetic algorithms have been used to solve many problems with very distinct natures. The genetic algorithms seek a solution that is "good enough" instead of "the best" solution.

The basic idea of genetic algorithms seeks to transform any solution of a problem into a "genetic code" that will be scored by an evaluation function, and when that score is not enough, the code is rejected. Explicitly, it is necessary to have a minimum number of generated genetic code solutions. This set of "genetic codes" is called a "population" and is used to evaluate against the problem. The expectation is that some of these codes are good enough for solving the problem. If not, some of the best codes are selected and recombined with the existing ones. The recombination here mentioned means performing an exchange of the scores of codes (depending on the type of solution), applying mutation (through a randomizing algorithm), and then reevaluating the new set of codes.

For IDS, these genetic algorithms may, for example, be used to acquire some knowledge of the examined rule-based system. The major difficulty encountered in using this approach lies in finding the right way to translate the solution into a genetic blueprint and defining the best evaluation function.

d) Biotechnology Algorithm

Biotechnology algorithms are used to detect masked attacks [43]. Masked attacks occur when the attacker succeeds in stealing a legitimate TCP session. The sequence alignment is a procedure used to determine the similarity between two DNA or protein sequences. The procedure can be executed globally, semiglobally, or locally by aligning the nucleotides or amino acids in each sequence and then producing a score indicating how well two sequences are aligned and, consequently, how similar they are.

The accuracy of DNA comparison and the flexibility provided by these algorithms have motivated their use in different applications. Particularly, the method has become attractive for solving intrusion detection problems because of conceptual equivalences between biotechnical matching and traffic pattern matching. For instance, the replacement of nucleotide sequences in biotechnology means the change of collections of user commands in intrusion detection prevention.

Although this method cannot offer the best system performance for intrusion detection in general, it particularly outperforms many other approaches when masked attacking is under discussion.

e) Fuzzy Logic Algorithms

Fuzzy logic is a computational methodology based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer logics are based. The fuzzy reasoning is based on rules and has enjoyed considerable successes dealing with problems in automation, control, data classification, decision analysis, and expert systems, among others. Based on the well-established fuzzy theory, the model for intrusion detection uses fuzzy logic to describe uncertainties and imprecisions of an intrusion attempt. This concept of uncertainty and imprecision is well accepted for the

case because security itself is relatively defined, so it can be uncertain and vague, thus fuzzy. The use of fuzzy logic algorithms in dealing with intrusion detection can increase the degree of reliability [44].

2.4 COMPUTATIONAL IDS TOOLS

To find out if a computer or a network has been violated is not a trivial or simple task. Sometimes, it is necessary to verify several pieces of information located at distinct units or processes inside an information system, such as the following:

- Log records (records of events)
- Unauthorized processes
- User accounts
- File systems change
- Others

Notice that the investigation of the related events and pieces of information definitely cannot be manually performed by an information system security manager. Instead, the verification and analysis should be carried out by appropriate and powerful computational tools. For network security, intrusion detection systems (IDS) are the desired and dedicated computational tools.

Under this context, the IDS can be described as an expert tool capable of reading and interpreting the contents of log files from routers, firewalls, servers, and other network devices, issuing a warning about a possible attack and identifying the hacker.

In this section, we will describe some IDS tools being widely used today. Most of these tools are based on the principle of signature matching.

2.4.1 Host-Based Intrusion Detection Systems (HIDS)

2.4.1.1 OSSEC (Open Source Security)

OSSEC [45] is an open source, host-based intrusion detection system developed by the Brazilian Daniel Cid with the following main functions:

- Logs analysis
- System integrity
- Rootkit detection
- Alerts and active responses (performed by firewalls or TCP wrappers)

OSSEC supports a large variety of logs, such as Unix pan, sshd (Open SSH), Unix telnetd, Samba, Su, Sudo, Proftpd, Pure-ftpd, vsftpd, Solaris ftpd, Imapd, and pop3d. Horde It can analyze protocols, search for specific content, and can also be used to detect a variety of attacks and probes, such as buffer overflows, portscans, CGI attacks, attempts to identify the operating system, among others. imp, Named (bind), Postfix, Sendmail, Iptables firewall, Solaisip filter firewall, AIX, ipsec/firewall, Netscreen firewall, Snort IDS, Apache web server (access log and error log), IIS web server, Squid proxy, Windows event logs, Generic unix authentication (adduser, logins, etc.). This HIDS offers the following operational modes:

- Local IDS: Monitoring and analyzing only the host on which OSSEC is installed
- Server: Monitoring and analyzing the logs sent by agents
- Agent: Working as a client and sending all information to the server for processing and analysis

OSSEC can work in conjunction with Snort [46] (a network-based intrusion detection system) under a hybrid detection environment, under which it analyzes the logs and alerts generated by Snort and classifies them according to the generated signatures and alerts the administrator whenever necessary.

2.4.1.2 Osiris

Osiris [47] is a host integrity monitoring system. It verifies any change occurring on network hosts as well as that on the file system and reports to the administrator. Osiris takes periodic snapshots of the file system and stores them in a database. This database, as well as the settings and logs, are stored on a central management server. When changes are detected, Osiris will record these events in the system log and optionally send an email to the administrator.

In addition to file monitoring, Osiris has the ability to monitor other system information, including user lists, group lists, and kernel modules or extensions.

Osiris runs on any Windows and UNIX system, including BSD, Linux, Mac OS X and Darwin, AIX, IRIX, and Windows NT/2K/XP. Such flexibility allows Osiris to manage any platform supported under a Windows or UNIX environment. Another facility that Osiris possesses is its modular interface. This unit allows developers to easily extend the monitoring functionality of scanning agents.

2.4.1.3 Tripwire

Tripwire [48] is a tool for evaluating the integrity of files, which is extremely useful for intrusion detection. Tripwire creates a database based on some information provided by system critical files, including file size and a cryptographic checking code. It compares the current information with that previously generated and detects the change. Then, it is the user's responsibility to decide whether this change was due to an attack or not.

In order to have reliable judgments, the database of Tripwire should be protected against forgery and invasion attempts. This can be done by keeping the database disconnected from the network or using a storage read-only mechanism. The configuration of Tripwire can be very complex for large multi-user systems. It is necessary to associate each file with its corresponding service or application and detect occurred changes.

2.4.1.4 HP-UX HIDS

HP-UX HIDS (Hewlett-Packard Development Company) [49] is a HIDS solution provided by Hewlett-Packard that performs monitoring and generation of an alert in the event of intrusion. The intrusion detection is based on areas of vulnerability, that is, when one area is investigated, audit data are correlated to determine which of them was exploited in near real time.

2.4.1.5 CACIC

CACIC is the first public IDS developed in Brazil [50] (from Portuguese – Configurador Automático e Coletor de Informações Computacionais). CACIC is able to provide an accurate diagnosis of the computational park and display information, such as the number of devices and their distribution in various units, the types of used and licensed software, and hardware configurations, among others. It can also provide standard information and the physical location of attached equipment devices, thus extending the control of the park computational and network security.

2.4.1.6 Nagios

Nagios [51] is open source software, created originally by Ethan Galstad and called Netsaint in the 1990s. Currently, there is a large community of programmers based around developing Nagios-based IDS tools. It is a popular network monitoring application with a distributed open source under the GPL license. It can monitor both hosts and services, alerting the user when problems occur and when problems are resolved.

2.4.1.7 Radmind

The Radmind [52] consists of a set of UNIX commands and a designed manager to execute evaluation on several other client machines. Radmind has Tripwire as its core and implements an additional tool enabling the recovery of the machine after having being changed.

2.4.1.8 Other HIDS Tools

In this section will be listed other examples of host-based intrusion detection system tools as well as their link to where it is possible to obtain more information.

- Advanced Intrusion Detection Environment (AIDE)
  http://sourceforge.net/projects/aide
- Another File Integrity Checker (AFICK)
  http://afick.sourceforge.net/
- AuditGUARD
  http://www.s4software.com/ag.htm
- Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD's eXpert-BSM)
  http://www.sdl.sri.com/projects/emerald/releases/eXpert-BSM/index.html
- System iNtrusion Analysis and Reporting Environment (SNARE)
  http://sourceforge.net/projects/snare/
- Enterasys Dragon Host Sensor
  http://www.enterasys.com/ids/
- GrSecurity – PaX
  http://www.grsecurity.net/
- LIDS
  http://www.lids.org/
- Logsurfer
  http://www.dfn-cert.de/eng/logsurf/
- McAfee Host Intrusion Prevention
  http://www.mcafee.com/us/enterprise/products/host_intrusion_prevention/index.html
- Samhain
  http://www.la-samhna.de/samhain/
- Sentry Tools
  http://sourceforge.net/projects/sentrytools/

2.4.2 Network-Based Intrusion Detection Systems (HIDS)

2.4.2.1 Snort

The Snort [46] is an open source network-based intrusion detection system, capable of performing traffic analysis and packet capture in real time on networks using the IP protocol. Snort has three operational modes, namely, the following:

- Packet capturing
- Network traffic analysis (Sniffer)
- Complete network system intrusion detection

Snort performs its detection based on signatures using a flexible rules language to analyze collected traffic data. These signatures/rules are updated daily by its development/management team as well as by enthusiasts and volunteers. The updating information and results can be acquired directly from the Internet, normally through some automated procedures.

Besides its established rules, Snort also interacts with those preprocessors located at network devices. These preprocessors are designated to perform some specific functions, which are crucial for Snort being efficient. Some well-known functionalities include portscans detection, complex attack pattern detection, and reassembling mechanisms of fragmented packet sequences.

Snort also interacts with Firewalls. The integrated mechanisms transform Snort into a reactive IDS model. Some important integrated mechanisms are SnortSam and Snort-Inline. SnortSam allows that SNORT interacts directly with many different network firewalls currently available in the market, ranging from simple and basic IPChains/IPTiptables to highly advanced Cisco firewalls.

Snort-Inline is a compiled version of SNORT. This integrated mechanism offers alternatives for dealing with input traffic packets. Instead of recording and alert generation in a suspicious connection, one may "drop" (delete) packets.

2.4.2.2 Internet Security System (ISS)

An Internet security system [53] is a real time IDS. Its architecture consists of three parts: a recognition mechanism based on the network (HIDS), a mechanism-based machine, and an administrator module.

The mechanism of network recognition is implemented on dedicated stations, each responsible for a network segment. This mechanism seeks to identify packets having attack characteristics. When an intrusion activity is detected, some action will be executed, for example, termination of the network connection, sending an alert, recording the session, reconfiguring the firewall, and sending an alert to the administrator module, among others. Mechanism-based machines compare input patterns with those recorded in order to identify attacks. The administrator module receives data from both detection mechanisms (recognition mechanism based on the network and mechanism-based machines). The presence of this module enables centralized configuration and system administration.

2.4.2.3 Kismet

Kismet [54] is a network analyzer (a sniffer) and an intrusion detection system for 802.11 wireless networks. When Kismet is implemented on wireless cards operating in the monitoring mode, the cards can capture network packets under different protocols, i.e., 802.11a, 802.11b, and 802.11g. Kismet also can run under the following operational systems: Linux, FreeBSD, NetBSD, OpenBSD, and Mac OS X.

2.4.2.4 Cisco Secure IPS

The Cisco Secure IPS [55] is a commercial IDS. This IDS actually is an updated version of NetRanger, one of the very first intrusion detection systems. The Cisco Secure IPS is a powerful IDS, capable of combining and executing different network security facilities and also acting as a prevention system against intruders.

2.4.2.5 Prelude IDS

The Prelude Hybrid [56] is an open source network-based intrusion detection system similar to Snort. The Prelude Hybrid is used to manage security information security and can be integrated with other network tools, including even more system intrusion detectors (sensors).

2.4.2.6 Bro Intrusion Detection System

The Bro Intrusion Detection System [57] is a system that performs passive monitoring of network traffic and analyzes the events with suspicious activities through semantic analysis, i.e., from the acquired knowledge, the IDS becomes enabled to investigate attacks.

2.4.2.7 Other Examples of NIDS Tools

In this section will be listed other examples of network-based intrusion detection systems tools as well as their link to where it is possible to obtain more information.

- Cyclpops IDS
  http://www.e-cop.net/cyclops-intrusion-detection-system.html
- Enterasys Dragon IDS
  http://www.enterasys.com/products/advanced-securityapps/dragon-intrusion-detection-protection.aspx
- Firestorm NIDS
  http://www.scaramanga.co.uk/firestorm/
- GFI LANguard
  http://www.gfi.com/lannetscan/?adv=62&loc=22&adclickid=14257624
- RealSecure Network
  http://www-935.ibm.com/services/us/en/it-services/gts-it-service-home-page-1.html
- SecureMetrics
  http://www.securitymetrics.com/securitymetricsappliance.adp
- Shoki
  http://shoki.sourceforge.net/
- SNIPS
  http://www.navya.com/software/snips/
- McAfee IntruShield Network IPS Solution
  http://www.mcafee.com/us/business-home.aspx
- SnorbySpsa
  http://bailey.st/blog/snorby-spsa/

2.5 CONCLUSION

The intrusion detection systems (IDS) have evolved considerably since the 1980s and are now indispensable in today's communication networks.

Modern intrusion detection systems are based almost exclusively on signature detection strategies due to their remarkable successes in detection of most already known attacks. However, the appearance of new and sophisticated attacking methods is constantly and quickly making network systems again vulnerable even though they are supposedly protected by the well-established signature detection systems. Although the new approach of anomaly detection has been invented to overcome this drawback, up until now, the reported performances are still far from satisfactory, precisely because of its high false alarm rates. Definitely further research on anomaly detection is needed.

As described in this chapter, many different IDS methodologies and tools have been proposed, implemented, and used by network systems. Unfortunately, individually they have their own weak points, which make them vulnerable. Fighting against those malicious attempts, which vary and are becoming more sophisticated day by day, requires more complex and intelligent IDS. A promising tendency toward this end currently focuses on new approaches that intelligently combine these methodologies.

Characterization of "normal" behaviors has also become one of the main stimuli in the field of intrusion detection systems. One expects that, based on some standardized behaviors, it is possible to easily detect anomaly behaviors. Unfortunately normal behaviors are environmentally dependent. In other words, optimization of anomaly detectors is not a trivial task due to environmental influences. Under this context, a possible challenge for new intrusion detection systems based on anomaly detection could be development of adequate theory for intrusive behavior in order to help detection system designing.

It is also noteworthy to observe that intrusion detection systems alone cannot offer all security purposes; the use of other types of protection mechanisms will always be necessary.

AUTHORS' BIOGRAPHIES

Jeferson Wilian de Godoy Stênico received a BS in mathematics from the Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP, São Paulo State University), Brazil (2006) and a MSc in electrical engineering from State University of Campinas – Unicamp, Brazil (2009). He is currently a PhD student in electrical engineering from the State University of Campinas – Unicamp. His current research interests include network traffic modeling, network design, performance analysis, and communications systems.

Lee Luan Ling received BS and MSc degrees in electrical engineering from the University of São Paulo (1980) and the State University of Campinas (1984), respectively, in São Paulo, Brazil. In 1991, he received a PhD degree in electrical engineering from Cornell University, Ithaca, United States. In 1984, he became a faculty member at the School of Electrical and Computer Engineering, State University of Campinas where currently he is a full professor. Dr. Lee has published more than 150 technical papers and also serves as a reviewer for many journals and international conferences. His current research interests include pattern recognition, handwriting recognition, biometrics, image processing, artificial intelligence, video monitoring and surveillances, network traffic modeling, and network design and performance analysis. Dr. Lee is the recipient of several awards, including the 1997 academically outstanding young Chinese in Brazil and Honorable Mention Award in the 1993 National Invention Competition (São Paulo, Brazil).

REFERENCES

1. Anderson, J. P. *Computer Security Threat Monitoring and Surveillance*, Fort Washington, 1980.
2. Denning, D. E. and Neumann, P. G. *Requirements and Model for IDES—A Real-Time Intrusion Detection Expert System*, SRI International, 1985.
3. Denning, D. E. An intrusion-detection model, *IEEE Transactions on Software Engineering*, Volume SE-13, Issue 2, Feb. 1987, pp. 222–232; also in *Proceedings of the 1986 Symposium on Security and Privacy, IEEE Computer Society*, April 1986, pp. 118–131.
4. Smaha, S. E. Haystack: An intrusion detection system, *IEEE Fourth Aerospace Computer Security Applications Conference*. TracorAppl Sci. Inc., Austin, TX, pp. 37–44, 1988.
5. Distributed Intrusion Detection System (DIDS), http://seclab.cs.ucdavis.edu.
6. Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., and Wolber, D. A network security model, In *Proceedings of the Symposium on Research in Security and Privacy*, Oakland, CA, pp. 296–304, 1990.
7. Proctor, P. Audit reduction and misuse detection in heterogeneous environments: Framework and application. In *Proceedings of the Tenth Annual Computer Security Applications Conference*, pp. 117–125, Orlando, FL, 1994.
8. Automated Security Measurement System (ASIM), http://www.access.gpo.gov.
9. 15th Annual Computer Crime and Security Survey, CSI/FBI, 2010/2011, https://cours.etsmtl.ca/log619/documents/divers/CSIsurvey2010.pdf.
10. Computer Crime and Security Survey, CSI/FBI, vol. VIII, no. 1, Spring 2002. http://diogenesllc.com/2002cybercrimesurvey.pdf.
11. IBM Emergency Response Service Security Vulnerability Alert ERS-SVA-E01-1996:006.1 Newly available patches for IBM AIX(r) address "SYN flood" and "ping of death" vulnerabilities, http://www.nmr.mgh.harvard.edu/MGH-UNIX/sec_advisories/CIAC:H-12.html.
12. Ranum, M. J. *Coverage in Intrusion Detection Systems*. NFRSecurity, Inc. Technical Publications, pp. 1–9, June 2001.
13. Shah, B. How to choose introduction detection solution. Whitepaper, SANS Institute, InfoSec Reading Room, July 2001.
14. Hochberg, J., Jackson, K., Stallings, C., McClary, J. F., DuBois, D., and Ford, J. Nadir: An automated system for detecting network intrusion and misuse. *Journal Computers and Security*, vol. 12, no. 3, pp. 235–248, May 1993.

15. Heberlein, L. T., Mukherjee, B., and Levitt, K. N. Internet security monitor: An intrusion detection system for large-scale networks. In *Proceedings of the 15th National Computer Security Conference*, Baltimore, MD, vol. 12, no. 3, pp. 235–248, 1993.
16. Hofmeyr, S. A., Forrest, S., and Somayaji, A. Intrusion detection using sequences of system calls. *Journal of Computer Security*, vol. 6, pp. 151–180, 1998.
17. Schupp, S. Limitations of network intrusion detection. SANS Institute, December 2000, http://www.sans.org.
18. Wang, Y. A hybrid intrusion detection system. PhD Thesis, Iowa State University, Ames, IA, 2004.
19. Nakamura, E. T., and Geus, P. L. *Segurança de Redes em Ambientes Cooperativos*. Novatec, 2007.
20. IETF, Internet Engineering Task Force, http://www.ietf.org.
21. IETF Intrusion Detection Exchange Format Working Group, http://www.ietf.org/html.charters/idwg-charter.html.
22. SNMP, http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol.
23. Park, J. A. Survey on flow-based internet traffic measurement technologies. Asian Info-Communications Council, Electronics and Telecommunications Research Institute (ETRI), Korea, 2005.
24. Cottrell, L. Network monitoring tools. Last updated: 2013, http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html.
25. Jacobson, V., Leres, C., and MacCanne, S. Tcpdump. Last updated: August 2009, http://ftp.ee.lbl.gov.
26. Combs, G. Wireshark. http://www.wireshark.org.
27. Munz, G. and Carle, G. Distributed network analysis using TOPAS and Wireshark. *IEEE Network Operations and Management Symposium Workshops*, pp. 161–164, April 2008.
28. Deri, L., and Suin, S. Effective traffic measurement using ntop. *IEEE Communications Magazine*, vol. 38, no. 5, pp. 138–143, May 2000.
29. CAIDA, cflowd Tools 1998, http://www.caida.org/tools/measurement/cflowd.
30. Plonka, D. Flowscan, http://www.caida.org/tools/utilities/flowscan/.
31. Dubendorfer, T., Wagner, A., and Plattner, B. A framework for real-time worm attack detection and backbone monitoring. In *Proceedings of the First IEEE International Workshop on Critical Infrastructure Protection*, November 2005.
32. Bin, L., Chuang, L., Jian, Q., Jianping, H., and Ungsunan, P. A NetFlow based flow analysis and monitoring system in enterprise networks. *Computer Networks*, vol. 52, Issue 5, pp. 1074–1092, April 2008.
33. Zhou, A., Yan, Y., Gong, X., Chang, J., and Dai, D. SMART: A system for online monitoring large volumes of network traffic. In *IEEE 24th International Conference on Data Engineering*. ICDE 2008, pp. 1576–1579, April 2008.
34. Cranor, C., Gao, Y., Johnson, T., Shkapenyuk, V., and Spatscheck, O. Gigascope: High performance network monitoring with a SQL interface. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 623–627, 2002.
35. Cranor, C., Johnson, T., and Spataschek, O. Gigascope: A stream database for network applications. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 647–651, 2003.
36. Plagemann, T., Goebel, V., Bergamini, A., Tolu, G., Urvoy-Keller, G., and Biersack, E. W. Using data stream management systems for traffic analysis—A case study. *Proceedings of the 5th International Workshop on Passive and Active Network Measurement* (*PAM '04*), pp. 215–226, 2004.
37. Ligocki, N., Gomes, C. L., and Hara, C. A flexible network monitoring tool based on a data stream management system. In *Proceedings of the IEEE Symposium on Computers and Communications* (*ISCC 2008*), pp. 800–805, July 2008.
38. Phaal, P., Panchen, S., and McKee, N. RFC 3176: InMon Corporation's sFlow: A method for monitoring traffic in switched and routed networks, September 2001. http://tools.ietf.org/html//rfc3176.
39. Phaal, P. and Panchen, S. Packet sampling basics [Online]. Available at http://www.sflow.org/packetSamplingBasics.
40. Duffield, N., Lund, C., and Thorup, M. Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, Marseille, France, pp. 159–171, 2002.
41. Sasha-Phrack Inc. Holistic approaches to attack detection. Volume 0x0b, Issue 0x39, Phile #0x11 of 0x12, 2003, http://www.phrack.com/issues.html?issue=57&id=11#article.
42. Li, W. Using genetic algorithm for network intrusion detection. In *Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference*, pp. 24–27, 2004.
43. Coull, S. E., Branch, J. W., Szymanski, B. K., and Breimer, E. Intrusion detection: A bioinformatics approach, In *Proceedings of the 19th Annual Computer Security Applications Conference*, pp. 34–33, 2003.

44. Dhopte, S., and Tarapore, N. Z. Design of intrusion detection system using fuzzy class-association rule mining based on genetic algorithm. *International Journal of Computer Applications*, vol. 53, no. 14, 2012.
45. OSSEC, http://www.ossec.net.
46. SNORT, http://www.snort.org.
47. OSIRIS, http://osiris.shmoo.com/.
48. Tripwire, http://www.tripwire.com.
49. HP-UX HIDS, http://h20338.www2.hp.com/hpux11i/cache/324806-0-0-0-121.html.
50. CACIC, http://www.softwarepublico.gov.br/spb/ver-comunidade?community_id=3585.
51. Nagios, http://www.nagios.org/.
52. Craig, W. D., and McNeal, P. M. Radmind: The integration of filesystem integrity checking with filesystem management. In *LISA '03: Proceedings of the 17th USENIX Conference on System Administration*, pp. 181–196, 2003.
53. IBM Internet Security Systems, http://www.iss.net.
54. Kismet, http://www.kismetwireless.net/.
55. Cisco Intrusion Detection, http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/index.shtml.
56. Prelude IDS, https://www.prelude-ids.org/.
57. Bro Intrusion Detection System, http://bro-ids.org/.