Anomaly Detection in Computer Networks: A State-of-the-Art Review

Sherenaz Al-Haj Baddar[1], Alessio Merlo[2]*, and Mauro Migliardi[3]
[1]*Department of Computer Science, KASIT, The University of Jordan, Amman, 11942, Jordan*
s.baddar@ju.du.jo
[2]*Computer Security Lab (CSec Lab), DIBRIS - University of Genova, Genova, 16145, Italy*
alessio.merlo@unige.it
[3]*DEI - University of Padova, Padova, 35131, Italy*
mauro.migliardi@unipd.it

Abstract

The ever-lasting challenge of detecting and mitigating failures in computer networks has become more essential than ever; especially with the enormous number of smart devices that get connected to all sorts of network everyday. Whether the root cause of a given anomaly is a security breach, a component failure, an environmental factor, or even any combination of these reasons, anomalies need to be detected and mitigated timely and properly. In this paper, we review and evaluate the state-of-the-art studies on the problem of anomaly detection in computer networks. We provide an elaborate description of the anomaly detection problem, and depict the different categorizations of its solutions. We also illustrate some recent state-of-the-art solutions on the network level, and depict current trends in handling malware-induced anomalies in smartphone networks. Additionally, we evaluate the presented solutions and highlight their shortcomings.

Keywords: Anomaly Detection, Network-level Detection, Application-level Detection, Mobile Security, Android Security

1 Introduction

Nowadays, multitude of interconnected computers, together with the abundance of smartphones, wearables, biomedical sensor, and the countless number of everyday appliances can connect seamlessly to different kinds of networks [1, 2, 3, 4]. Thus, the ever-lasting challenge of detecting and mitigating failures has become more essential than ever, whether it is a small piconet at home, a corporate network, or even the Internet.

A random, nonconforming, or unexpected behavior within a system is referred to as an anomalous behavior, or simply an anomaly [5, 6, 7, 8]. The existence of anomalies can have profound effects on the robustness of a given network's operation [9]; as such, the consequences of failing to detect anomalies can be dramatic. Anomalies going unnoticed in a given system may leak confidential information to the outside world [10, 11], cause financial complications and/or losses [12, 13], or, even worse, lead to making fatally wrong decisions [14]. While the existence of security breaches is usually manifested by anomalous behaviors [15], typical hardware and/or software malfunctions can result in anomalous behaviors as well [16, 17, 18]. Either way, anomalies within a computer network cannot go unnoticed, they need to be detected, located, and mitigated timely and properly.

Anomalous behaviors are in essence Byzantine failures. According to his seminal paper [19], Lamport

*Corresponding author: Computer Security Lab, DIBRIS, University of Genova, Viale F. Causa, 13, 16145, Genova. Tel: +39-010-353-2344. Homepage: `http://www.csec.it/people/alessio/`

stated that a system of interconnected nodes is said to have Byzantine failures when some node(s) send(s) conflicting messages to different parts of the system. A solution of the Byzantine failures should seek to establish an agreement between the non-faulty nodes in the system, taking things a step further than simply detecting that some nodes are being faulty.
Over many decades, the problem of detecting and mitigating anomalous behaviors has been studied thoroughly, yet it remained one of the most chronic problems in distributed computing [20, 21, 22, 23]. As technology evolves, new issues arise that add to the pile of unsolved issues pertaining to anomaly detection [24, 25]. In this paper, we aim at reviewing the state-of-the-art solutions of the anomaly detection problem in computer networks. Classically, detecting anomalies in a computer network, either wired or wireless, comprises inspecting the exchanged packet headers and/or contents. However, as smartphone technology emerged, networks of interconnected smartphones have become prone to different classes of anomalies. Thus, anomaly detection in computer network can be performed on two levels; the network level and the application level. Here, we divide our discussion into two parts: the first pertains to classical anomaly detection at the network level, while the second pertains to anomaly detection in smartphone networks at the application level.

Solving the anomaly detection problem is far from trivial, as the nature of anomalies themselves is varying. In the context of a computer network, providing a comprehensive definition of an anomalous or even a normal behavior is typically subtle [26, 27]. Another reason is that several anomaly detection techniques require labeled samples of normal and/or anomalous behaviors which are not easy to obtain [28, 29]. Besides, choosing a suitable tool to help detect anomalies is not straightforward. A designated tool may be well-suited for one kind of anomalies, but not for others [30]. Thus, when the types of anomalies are not known a priori, which is a rather realistic assumption, selecting an anomaly detection technique is not trivial. In the context of wireless and smartphone networks, nodes usually wander freely and could join and leave the network arbitrarily. Thus, the network dynamics impose further complications on the anomaly detection solution. Moreover, the network scale is an issue: anomaly detection solutions need to account for load-balancing and fault-tolerance themselves, especially with the growing sizes of current networks [31, 32].

The rest of this paper is organized as follows; Section 2 presents necessary terminology and background on the topic of anomaly detection. It also presents several categorizations of anomaly detection techniques. In Section 3, some state-of-the-art network-level and application-level anomaly detection solutions are discussed. A summary of the major shortcomings in the illustrated solutions is depicted in Section 4. Finally, the drawn conclusions are presented in Section 5.

2 Preliminaries and Background

The solutions of the anomaly detection problem originate from a wide array of disciplines. Thus, we introduce the basic concepts and terminology related to these disciplines, and shed light on relevant background. We also depict several categorizations of the techniques employed to solve this problem.

2.1 Basic Terminology

Anomalies are detected by analyzing the system's events, where each event is designated by a data instance, or simply an instance. The data instance possesses features (i.e. attributes) to help describe it [26]. For example, in network-level anomaly detection, a packet sent from a source node to a destination node may count as an event to be analyzed. Hence, the features of the data instance that corresponds to this event would include the packet's source address, destination address, length, and the time at which the packet was sent, alongside other features. It is worth mentioning that a data instance that has a single feature is referred to as a univariate instance, while an instance that has multiple features is referred to as a multivariate instance [30].

Features are crucial for distinguishing normal behavior from anomalous behavior. A given computer network typically provides a wealth of features per a given data instance, yet they are not necessarily all equally informative [33]. Thus, identifying the subset of the features that do actually separate normal behavior from anomalous behavior is not trivial. Some researchers utilized information theoretic approaches to help distinguish informative features [34, 35, 36, 37]. In information theory, the term entropy measures the uncertainty in the distribution of a given set of data [38]. Based on the concept of *entropy*, the Information Gain (IG) and chi-squared methods can be applied to help select the most informative features. The IG method measures the reduction in entropy when the feature is kept instead of being eliminated [39]. Also, the chi-squared test measures the independence of any two features, and thus helps retain the less correlated features [40]. A third method is the Fisher-score, which is the ratio of inter-class variance to the intra-class variance for a given feature, assuming that normal instances form a class and anomalous instances form another class [41]. In a third approach, researchers used machine learning classification techniques to achieve the same goal [42]. In another approach, the researchers would hand-pick the informative features to help detect anomalies [43].

2.2 Taxonomies of Anomaly Detection Techniques in Computer Networks

As illustrated is Section 1, an anomaly detection solution (i.e. detector) in networks operates either at the network-level or at the application-level. Aside from this categorization, various surveys and reviews provided different categorizations of anomaly detection techniques [30, 26, 33, 44]. In this subsection, we highlight the most commonly-referenced categorizations of anomaly detection techniques.

2.2.1 According to Granularity

When an independent data instance comprises an anomaly, it is referred to as a point anomaly [30, 26, 33]. For example, a system event in which a user tries to access a restricted server is considered a point anomaly. Many anomaly detection solutions assume this kind of anomalies [30]. Yet, point anomalies do not fit the situations where anomalous behavior is an aggregate of data instances, or when anomalies are associated with given contexts. A pictorial illustration of a point anomaly is illustrated in Fig. 1.

In other situations, the anomalous behavior is defined within a context; thus, a given data instance is not anomalous unless it happens within a predefined context [30, 33]. This is referred to as contextual anomaly, where the data instance has to have some feature(s) that pertain to the context, whether it is temporal (i.e. time-relevant), spatial (i.e. location-relevant), or a different kind of context per the problem domain. For example, having a staff member attempt to log in to her corporate system using her credentials is not anomalous per se. However, when this does not happen within the pre-defined business hours, the instance becomes anomalous in the temporal context. Fig. 2 illustrates an example of a contextual anomaly.

A collective anomaly designates a group of instances that exhibits an anomalous behavior compared to the other groups of instances [30, 33]. An individual instance within the anomalous group is not necessarily anomalous on its own. For example, consider a system in which users interact with a web server, where a supposedly non-privileged user does the following sequence of operations:

access a local folder on the server, upload an executable file into the folder, run a script

While typical non-privileged users only access the public folder(s) in the web server to simply read and upload non-executable files; this particular user's activities are nonconforming, and designate a collective anomaly. A pictorial example of a collective anomaly is depicted in Fig. 3.

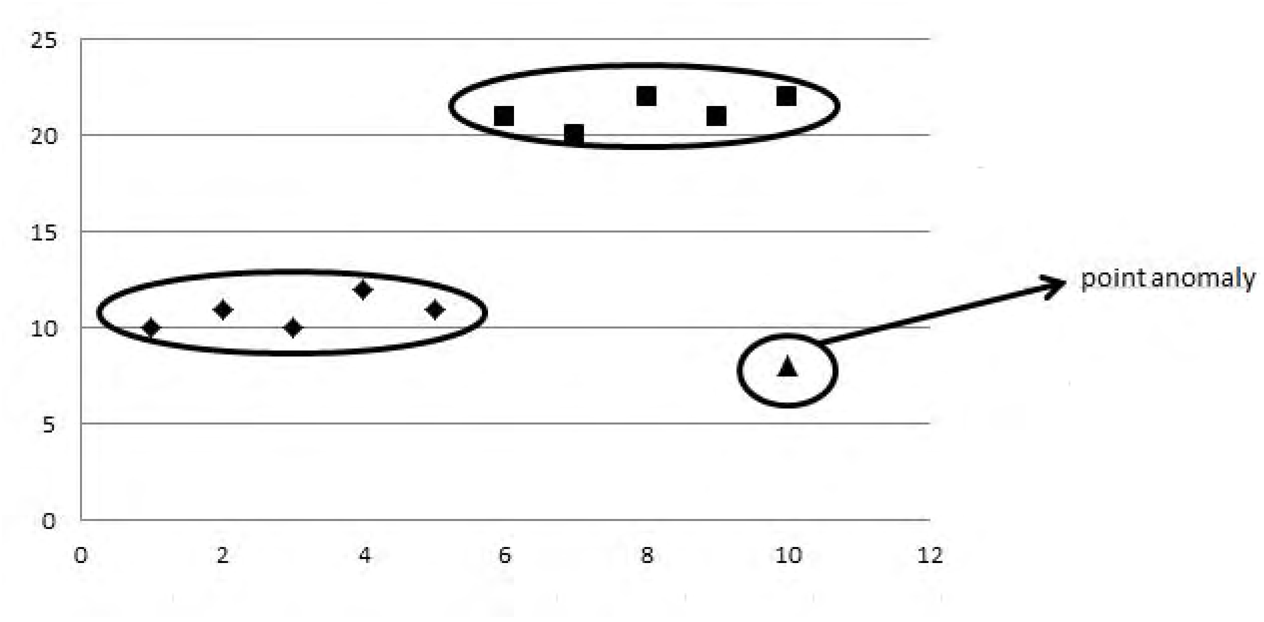

Figure 1: An example of a point anomaly.

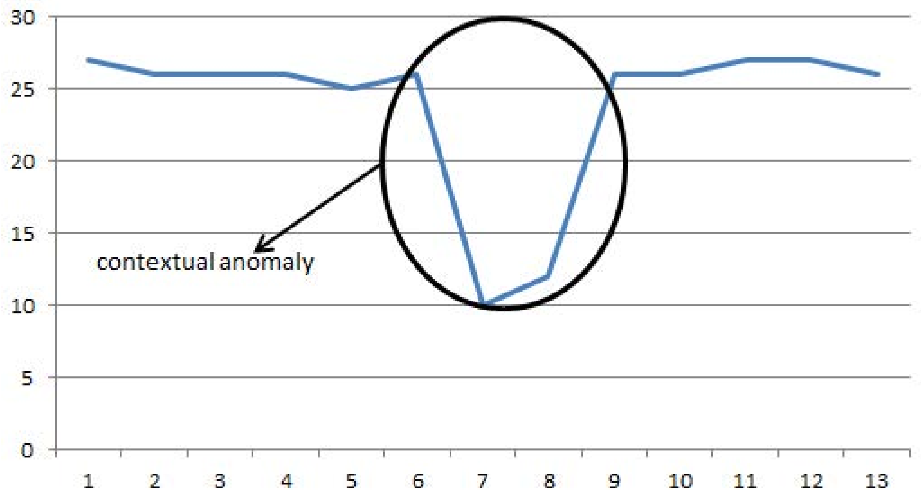

Figure 2: An example of a contextual anomaly.

2.2.2 According to Functionality

Typically, an anomaly detection solution is either signature-based [45] or behavioral [33, 37]. Signature-based solutions operate by applying a set of hardwired patterns, signatures, or rules against given behavior(s). If a given behavior matches either one of the hardwired signatures, then an anomaly is detected. Otherwise, the detector will not tell whether or not the designated behavior is anomalous. Typical examples of signature-based anomaly detectors are antivirus engines [46, 47]. Albeit being potentially efficient in terms of computational cost, these solutions fail at identifying new or previously unseen anomalies. Thus, their application domains are rather restricted.

Behavioral detectors, on the other hand, can learn the normal and/or anomalous behavior(s) of a

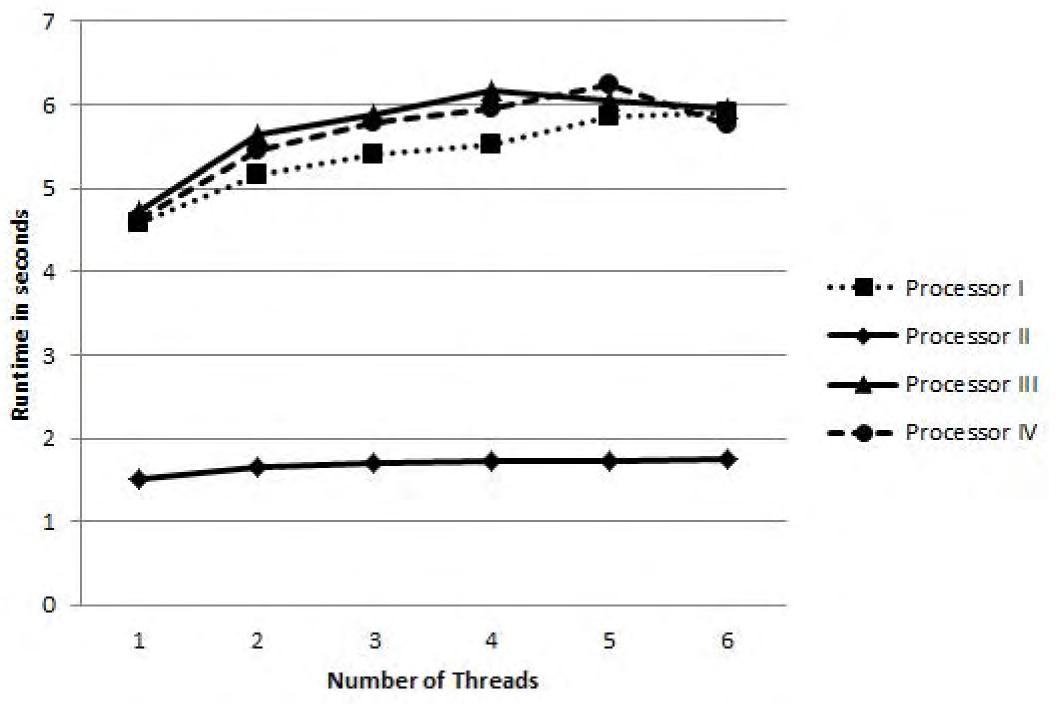

Figure 3: An example of a collective anomaly illustrated by the values of Processor II.

network and, thus, have the potential to identify whether new or previously unseen behavioral patterns are anomalous or not [37]. Such solutions would typically utilize one or more of the approaches described in subsection 2.2.3 in order to achieve their goal. Thus, they may experience longer computational times. Moreover, they are prone to detection inaccuracies which could also limit their applicability in some real-world setups.

2.2.3 According to the Underlying Approach

Anomaly detection can be considered a classification problem, where anomalous behaviors need to be distinguished from normal behaviors. Here, we discuss the taxonomy of the approaches used in anomaly detection solutions as depicted in Fig. 4, based on the discussions provided in [30, 48, 49].

The Statistical Approach. The statistical approach is used when normal data exist in high probability regions of some stochastic model, whereas anomalies occur in the low probability regions of that model [30]. The stochastic model is either determined a priori (i.e. parametric modeling), or is derived from the data itself (i.e. non-parametric modeling). The main idea behind this approach is to measure the anomaly score which designates the deviation of a given data instance from the model. Then, the score is compared to some pre-defined threshold (i.e. tested) to determine whether its corresponding data instance is anomalous or not. Hypothesis testing and decision theories are usually used to accomplish this task [49, 26].
Parametric modeling spans Gaussian-based models, regression-based models, and mixtures of models [30]. In the Gaussian-based model, the data is assumed to belong to some Gaussian distribution. The model's parameters are calculated using the maximum likelihood estimation based on the data instances themselves [50]. Several tests can be used to determine whether or not a data instance is anomalous.

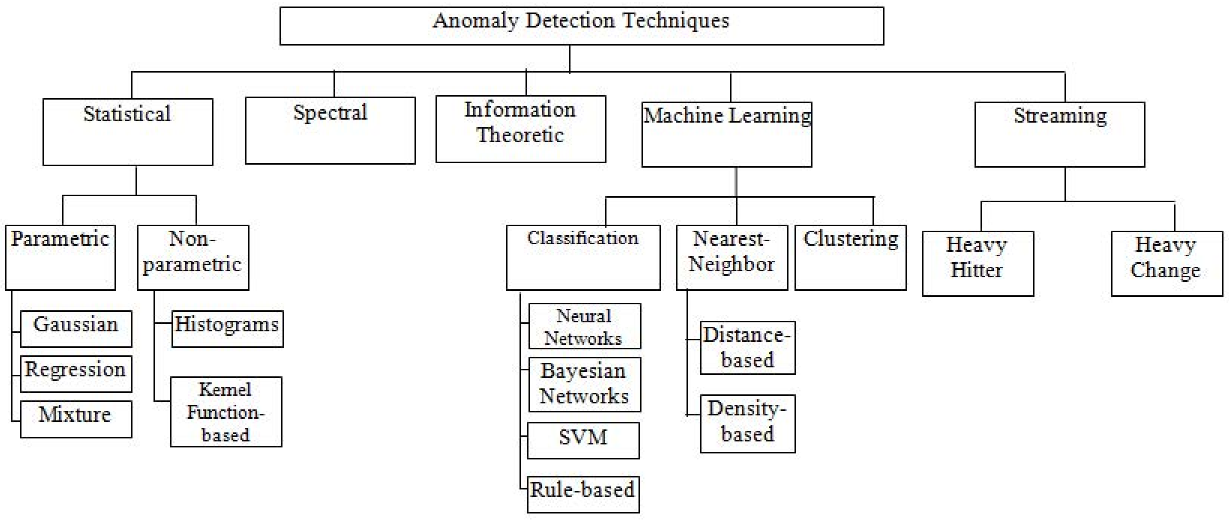

Figure 4: Taxonomy of anomaly detection techniques based on their approach.

Such tests include the box-plot[51], and the chi-squared tests. In the regression-based models, the data is fit to a regression model, like the linear model depicted in Fig. 5, and then the residual of each data instance, which is not covered by the regression model, gets measured. This residual is considered to be the instance's anomaly score. In mixture-based models, a combination of parametric models is utilized in two flavors. In the first flavor, two models are used: one to represent the normal data, and the other to represent the abnormal data. In this case, the test aims at determining to which distribution a given data instance belongs. In the second flavor, a mixture of parametric models can be used to represent the normal data only. Here, the test identifies a given data instance as anomalous if it does not belong to any of the designated models. For this approach to be effective, the model(s) to which normal and/or abnormal instances belong need to be determined a priori; the definition of these models is not always a trivial task.
In non-parametric modeling, no model is assumed a priori; instead, a model is generated from the normal data instances. Then, the deviation of a given data instance from the model would designate its anomalous score. This approach requires that normal behaviors are known a priori.

In the histogram-based model, the data is used to plot a histogram that comprises bins generated from the values of the normal data. Then, to determine whether a given instance is anomalous or not, it gets plotted to see if it falls in any one of the histogram bins. If this is not the case, the instance is considered to be anomalous. When this technique is used, the sizes of the bins are crucial: too wide bins would result in lower detection accuracy; while too thin bins would result is higher false alarms rates.
The modeling technique based on Kernel functions aims at deducing a similarity function based on the provided data; this allows constructing a model based on the data instances [52]. If the given data instances do not fully describe the designated behavior, the model will fall short in terms of accurate behavior identification.

The Spectral Approach. In some situations, several dimensions (i.e. features) of the data instances are inherently dependent. Thus, combining the dependent dimensions both improves the classification accuracy and reduces the computational complexity; the application of such a combination transforms

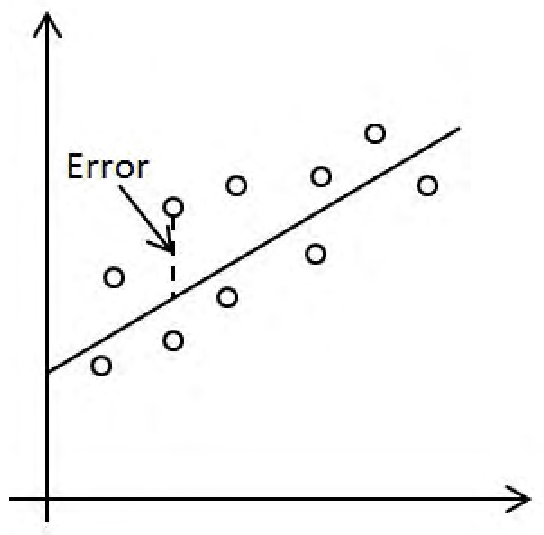

Figure 5: A set of points fit to a linear regression model.

the original data instances into new instances with only the independent dimensions.
Mathematically, this formulation is referred to as dimensionality reduction [53]. Formally, this reduction can be viewed as embedding the original data instances into a subspace with fewer dimensions as illustrated in Fig. 6; this is subject to the condition that normal and anomalous instances look drastically different after the dimensionality reduction has taken place. In the spectral approach, a dimensionality reduction algorithm is first applied to the data instances. One popular dimensionality reduction technique is the Principle Component Analysis (PCA) algorithm which gets applied to a matrix of the original instances, and generates a set of m orthogonal (i.e. independent) vectors referred to as principal components. The first k of the m vectors, where $k \leq m$, capture the highest variance in the original matrix and designate normal traffic (i.e. the normal subspace); the remaining $m-k$ vectors designate the anomalous traffic (i. e. the anomalous subspace). Nevertheless, the $m-k$ vectors can be further split into several noise as well as anomalies classes. Hence, to determine whether a data instance is normal or anomalous, it gets projected onto the normal and anomalous subspaces. If it is expressed more by the normal subspace, then it is considered normal; otherwise, it is considered anomalous [30]. The threshold used to choose either subspace is a separate parameter than needs to be adjusted properly.

The Information Theoretic Approach. From an information theoretic perspective, the data instances are considered to be a set of symbols generated by the network, whereas each instance is generated independently with a certain probability. Thus, one would seek to measure the average amount of information conveyed by each instance. This approach utilizes the concept of entropy, as described in subsection 2.1. It is based on the assumption that anomalies distort the information content of the network's data instances. Thus, the anomaly detection technique needs to split the data instances to subsets, so as to minimize the entropy. Several information theoretic techniques have been used in anomaly detection [54, 34, 35, 37], one of which is information gain as described in subsection 2.1. In this context, the IG method splits the instances into normal and anomalous to reduce entropy. Another technique is relative entropy which aims at measuring the statistical distance between two distributions [55].

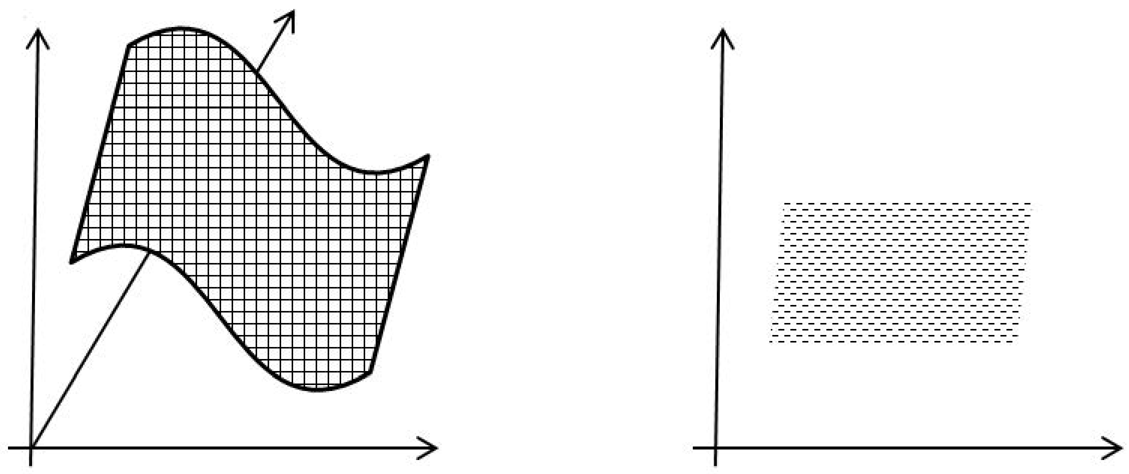
Figure 6: Dimensionality reduction of a high dimensionality data.

The Machine Learning Approach. One of the most appealing features of machine learning algorithms is that they improve their ability to distinguish normal behavior from anomalous behavior with experience [39]. In the context of anomaly detection, a machine learning algorithm typically provides a mapping that adapts to unseen network anomalies [49]. Formally, let set S be defined such that $S = \{0,1\}$, and let us assume that a data instance can be either in one of two states, normal (i.e. 0), or anomalous (i.e. 1). Let us also assume that each data instance resembles a feature vector X measured at time t and denoted by $X(t)$.
A machine learning algorithm aims at learning the function that maps all $X(t)$ instances to their proper states from S. In order to achieve their goal, machine learning algorithms utilize a set of data instances that resemble the instances within a given computer network; this set is referred to as a *training dataset* (or simply training set). Some machine learning algorithms learn the mapping function by utilizing labeled training sets, where each instance in the training set is labeled with either one of the states in S. These algorithms are referred to as supervised learning algorithms. On the other hand, some machine learning algorithms utilize training sets of totally unlabeled instances. These algorithms are called unsupervised learning algorithms. In some cases, the two approaches are mixed into the hybrid approach called semi-supervised learning, where the algorithm is trained with a majority of unlabeled instances and a minority of labeled instances. A machine learning algorithm starts by learning the mapping function from the training dataset, then proceeds to the testing phase, where it examines "other" data instances that are collectively referred to as the *testing set*, and computes the label (i.e. either 0 or 1) for each instance using the mapping function it learned.

According to the categorizations provided in [30, 26, 33], machine learning algorithms are either:

- **Classification-based**. These algorithms' main goal is to assign each data instance to either one of pre-set classes based on their features. Typical examples include:
  - *Classification-oriented neural networks*. A neural network loosely mimics the human neuronal structure, and comprises a set of highly interconnected processes that operate asynchronously on their local data [56]. A neural network is trained on normal data instances. After that, it is presented with unseen instances. Here, the network applies a test on the test data instance, if it passes, then it gets accepted as a normal instance. Otherwise, it is consid-

ered anomalous. Feed-forward networks are neural networks typically used in classification, like multilayer perceptron networks [57]. Depending on the labeling of the data, neural networks can be used for both supervised and unsupervised learning. Fig. 7 depicts a multilayer perceptron network.

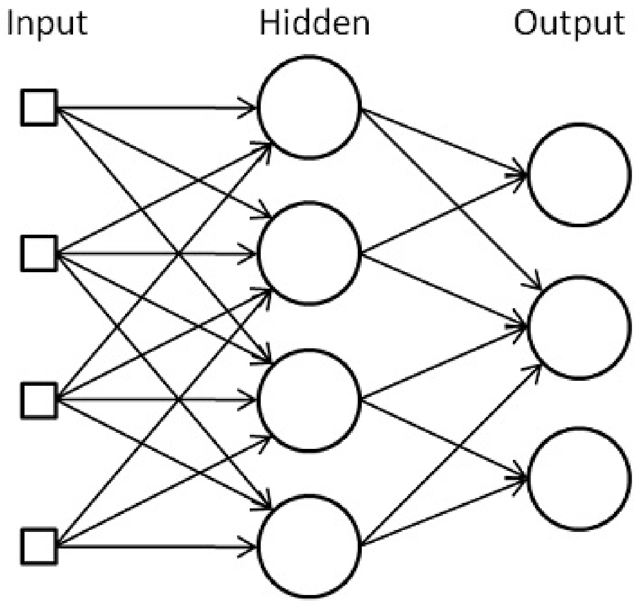

Figure 7: A multilayer perceptron network with one hidden layer.

- *Bayesian networks*. According to the definition in [58], "*a Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest*". Bayesian networks are supervised learning algorithms based on the well-known Bayes Theory [39]. They operate by estimating the posterior probability of an event given some pre-condition. A special class of Bayesian networks is referred to as Naïve Bayesian networks which are used for univariate categorical data instances [39]. Here, for a given data instance, the network estimates the posterior probability of detecting a class label from a set of normal and anomalous class labels. The class label with the largest posterior probability is selected as the class to which the data instance belongs. Multivariate data instances are handled via generalizing the univariate model, as the posterior probability for each attribute is estimated, then the estimated probabilities get combined to assign the data instance to a given class [30].
- *Support Vector Machines (SVM)*. SVMs are supervised learning algorithms that plot the training data instances in a multi-dimensional plane and then determine a hyperplane that splits the data instances into two disjoint groups while maintaining the maximum margins around the separating hyperplane [59]. One-class SVM algorithms are trained only with normal data. Thus, upon receiving a test data instance, they predict whether it belongs to the normal data class, or not. SVMs are well-defined as they stem from a solid mathematical background of statistical learning theory [59]. An SVM algorithm is considered a linear classifier when it uses a line to split the data instances into normal and anomalous. To perform non-linear classification, SVM algorithms use kernel functions [60]. An example of an SVM hyperplane is presented in Fig. 8.
- *Rule-based machine learning algorithms*. These supervised learning algorithms learn the rules that capture the normal behavior of a data instances. Thus, during testing, when all the

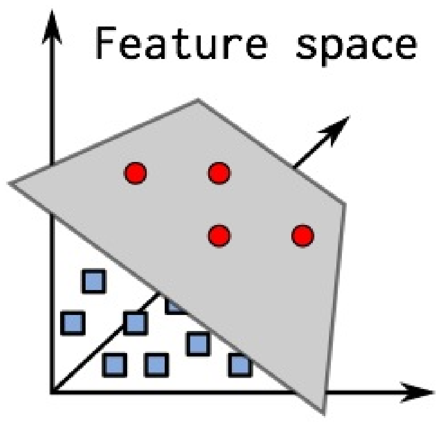

Figure 8: An SVM hyperplane splitting two sets of data.

rules fail to capture a data instance, it is considered anomalous. Decision trees and Association Rule Mining (ARM) techniques, among other rule-based techniques, are used to learn the rules from the training data instances [39, 30]. A well-known decision tree algorithm is the C4.5 algorithm, which is also known as J48 algorithm [39, 61]. Each rule is assigned a weight that is proportional to the ratio of the number of training data instances the rule classified correctly to the total number of training instances covered by the rule. For a given test data instance, the rule that best captures the test instance is sought. Here, the anomaly score is the inverse of the weight associated with the best rule. Random forests are constructed out of several decision trees; a random forest reports the mode of the classification of all individual decision trees as the overall classification result [62].

- **Nearest-neighbor algorithms**. These algorithms use either distance-based or density-based functions to measure the distance between a given data instance and its $k-th$ nearest neighbor [30]. This distance designates the anomalous score of that instance. The assumption is that normal instances, unlike anomalous instances, occur in dense groups. Based on whether labels are used in training data instances, these algorithms can either operate in supervised or unsupervised fashions.
- **Clustering**. These unsupervised learning algorithms operate by trying to identify groups (i.e. clusters) of closely located (or similar) training data instances. Anomalies may form sparse clusters or belong to no cluster at all. Self-Organizing Maps (SOM) [63], Expectation Maximization (EM) [64], k-means clustering [65], and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithms [66] are classical clustering algorithms. A pictorial example of the k-means algorithm output is shown in Fig. 9.

The Streaming Approach. The main idea behind this approach is to consider the data instances within the network as continuous flows, and try to extract information from almost every such flow. Thus, discrete algorithms may be applied to these streams in order to help detect anomalies [49]. The point is trying to avoid sampling of traffic, which is usually used in typical anomaly detection algorithms, as it may result in missing important data instances that may convey significant information on the normal or anomalous behavior within the network.
To do so, the problem of detecting anomalous behavior in a network can be reformulated into either [49]:

1. *Heavy-hitter detection problem*, where the anomaly detection technique aims at identifying flows

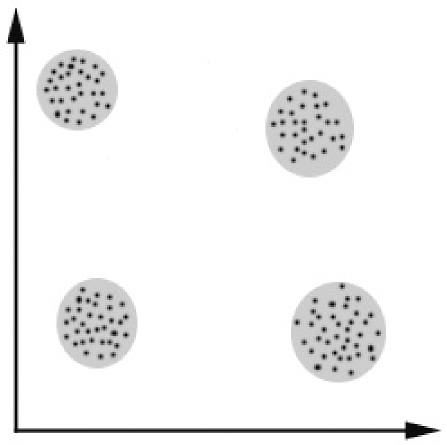

Figure 9: The output of applying k-means algorithm to a dataset with 4 clusters.

that represent large portions of the traffic;

2. *Heavy-change detection problem*, where the anomaly detection technique aims at identifying flows that represent a significant change in traffic volume from one period of time to the other.

The heavy-hitter problem might be of more interest to network service providers as they try to achieve their quality of service goals. The heavy-change detection problem, on the other hand, better models the anomaly detection problem in computer networks. To solve the heavy-change problem, one would typically [49]:

1. Apply a summarizing algorithm to the traffic flows to extract vital information without investigating the flows thoroughly. These are typically called sketching algorithms, and their outcomes are called sketches.

2. Apply time-series forecast models on top of the sketches.

3. Estimate forecast errors, and use them to identify whether significant changes in traffic volumes were spotted.

In the context of computer networks, let us assume that we aim at detecting heavy-change in each flow designated by its source IP address. Also, let the vector $F(t)[p]$ denote the amount of traffic transmitted from the node with IP address p during time interval t. If node p transmits δ additional bits during the t^{th} time interval, then the total amount of traffic associated with p becomes $F(t)[p] = F(t)[p] + \delta$. Now, to identify flows with heavy change, we need to find all flows q for which $|F(i)[q] - F(j)[q]| > \varepsilon$, for some predefined threshold ε at time intervals i and j. The difference can be simple absolute difference or a relative difference, or a variational difference [67]. Naturally, the streaming technique is useful in the situations where anomalous behaviors comprise notable changes in network traffic volumes.

2.2.4 According to the Nature of Data Instances

While some anomaly detection solutions rely on digesting the packet headers to help identify anomalies, others would analyze the packets' payload itself. Choosing either method depends on several factors. First, in some cases the anomalous behavior manifests itself in anomalous traffic patterns, regardless of the content being exchanged. For such anomalies, detecting packet headers would suffice, and help avoid breaching users' privacy. On the other hand, some anomalies do not manifest themselves in anomalous network traffic, but in falsified packets' payload. Here, inspecting the traffic headers would have little to do, if at all, with accurate identification of anomalies in the network. Yet, breaching users' privacy may be simply unacceptable at times.

2.3 Other categorizations

Anomaly detection solutions in networks are either packet-based or flow-based. In packet-based anomaly detection, the detector analyzes the traffic packet per packet in order to look for anomalies. Yet, as the number of such packets would be enormous, these solutions revert to sampling. Using sampling, the detector investigates some of the packets generated at predefined timeslots, and associated with previously-determined locations within the network. One challenge associated with sampling is its granularity, if sampling happens more often than it should, it would result in having the detector examine a larger number of packets which could render the detector rather slow in identifying anomalies. On the other hand, if sampling happens less often than it should, it would result in missing important information which could result in failing at detecting anomalies as well. In flow-based anomaly detection, the solution analyzes flows of traffic, running between given source and destination nodes, rather than individual packets. Again, the detector may either investigate every such flow in order to detect anomalies, or may sample certain flows to investigate. As we have shown in Subsection 2.2.3, streaming anomaly detection techniques use the concept of sketching algorithms to analyze summaries of traffic flows.

While some anomaly detection solutions monitor live nodes' behavior, and report the anomalies they identify in real-time (i.e. online detectors), other solutions operate offline, as they digest previously-collected behavior patterns. Online solutions are more responsive compared to offline solutions, yet, some of them may introduce some overhead on the network itself. This is due to the fact that some online solutions rely on having some or all of the nodes in the network participate in the anomaly detection process. This can be a serious burden in situations where nodes have limited resources such as the case of sensor and smartphone networks. Offline solutions, on the other hand, relief the network from such overheads, but fail at being responsive which may not be acceptable in some contexts.

The aforementioned categorizations of anomaly detection solutions are applicable both in network-level detectors as well as application-level detectors. However, other categorizations are more relevant to application-level anomaly detection solutions. For example, such anomaly detection solutions are considered static if they identify anomalies solely via inspecting the application's source code files. In the context of smartphone networks, these solutions would analyze the application's xml and .dex files in order to identify anomalous behaviors [37]. They would normally look for signs of anomalies in the application's permissions, used APIs, and the like. On the other hand, dynamic anomaly detection techniques identify anomalous behaviors by monitoring the application's behavior under execution, and normally look for signs of dynamic loading, suspicious code downloads, suspicious system calls, and the like. While static analysis is usually more efficient from a computational perspective, it fails when the application's source is obfuscated as it is usually the case for malicious code. Dynamic techniques, on the other hand, are more resource and time consuming, relatively speaking, which may render them impractical to deploy on smartphones and sensors. However, they are immune to code obfuscation.

Another categorization that especially pertains to smartphone and sensor networks is the location of the detection solution [68]. Some solutions run totally on-device without support from remote server(s) (see e.g. [69]). Such solutions need to be lightweight and efficient, or they will exhaust the devices' resources. Other solutions run off-device, where some or all of the detection processing takes place on remote server(s) that do the actual processing of the collected information, and decide on the existence of anomalies. These solutions impose a smaller computational overhead on the devices, yet they incur communication costs, and are prone to malicious isolation from the servers which will prevent them from communicating their measurements during anomaly detection.

2.4 Measuring the Performance of an Anomaly Detection Technique

As we briefly described in Subsection 2.2.3, anomaly detection starts by a learning phase in which the solution, regardless of its approach, gets exposed to the training dataset. After the learning phase is over, the solution becomes ready to classify new and/or previously unseen instances. Upon completing this phase, specific metrics that measure the solution's performance get calculated accordingly.

For a typical anomaly detection solution, identifying anomalies correctly is necessary, but not sufficient. It is also necessary that the solution does not generate false alarms (i.e. identifies normal instances as anomalies). In real-world applications, it is rather undesirable to have a solution that generates false alarms even if it manages to detect actual anomalies with high accuracy. Here, we summarize the most important metrics used to assess the performance of an anomaly detection solution.

The most-commonly used metric is the True Positive (TP) rate (also known as detection rate, sensitivity, or recall) which designates the ratio of the instances classified as anomalous (i.e. positive) by the detector. Another important metric is the False Positive (FP) rate (i.e. false alarm) which designates the ratio of the normal instances classified mistakenly as anomalous. The False Negative (FN) rate is the ratio of the anomalous instances mistakenly classified as normal, while accuracy refers to the ratio of all correctly classified instances either normal or anomalous. Moreover, the Receiver Operating Characteristic (ROC) curve is a metric to measure the performance of a detection solution. It is generated by plotting the TP rate against the FP rate at different threshold values, where a threshold is the cut-off point between considering an instance normal or anomalous. The AUC metric designates the Area Under the ROC Curve. Another metric is precision, which is the ratio of correctly classified anomalies to all instances classified as anomalies. F-score (also known as F-1 score, F1-score, or F-measure) is the harmonic mean of precision and recall and is calculated as [70]:

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{1}$$

3 Anomaly Detection Techniques: State-of-the-Art

In this section we discuss a set of recent anomaly detection solutions for computer networks. We start with network-level anomaly detection solutions, and then depict application-level anomaly detection solutions.

3.1 Network-level Anomaly Detection Solutions

Several network-level anomaly detection solutions have adopted the machine learning approach. Here, we depict a subset of these solutions.
In [71], the goal was rapid and accurate detection of some data-centric anomalies while saving on nodes' computational, memory, and communication resources. The authors introduced the Practical Online aNomaly Detection (POND) framework which utilizes machine learning for anomaly detection in Wireless Sensor Networks(WSNs). The WSN had a base station, and comprised a logical tree of nodes that extracted statistical features online without maintaining sampled data points over the extraction window. To reduce the communication overhead, the nodes sent summarized statistical measurements only during their initial deployments. Furthermore, child nodes sent their summarized measurements to their parent nodes to help them detect misbehaving child nodes. POND utilized supervised learning where certain anomaly models were injected into the sensor network initially. Then, each sensor measured three statistical features pertaining to temporal deviation, spikes in values, and violations of data ranges. After

the sensors communicated their statistical feature summaries to the base-station, it applied the Adaptive Boosting (AdaBoost) algorithm to the statistical data collected from sensors, and selected a predictor that achieved the best trade-off between complexity and accuracy. The AdaBoost algorithm operates by combining weak classifiers into a weighted sum that represents the overall classification value [72]. The predictor was sent to the sensor nodes, with normal and abnormal nodes labeled accordingly. A parent node would restrain from sending the statistical measurements of its child node(s) that are designated abnormal by the predictor. The authors did two limited sets of experiments; using simulation of 200 nodes, and implementation on a randomly scattered 22-node in-door network of MicaZ motes, and measured FN and FP rates.

In [73], the author proposed an anomaly detection algorithm that combined both a genetic algorithm and an artificial immune system, called GAAIS, for dynamic intrusion detection in ad hoc networks. GAAIS can adapt to changes in network topology and utilizes two updating methods to achieve this goal. GAAIS comprises three steps: training, detection, and updating. In the training step, each node extracts a set of feature vectors from its normal traffic to generate its spherical detectors. In the detection step, each node extracts a set of feature vectors from its current traffic and compares it, after proper scaling, to its spherical detector to help identify anomalies. In the updating step, the spherical detectors get updated using either partial or total update at pre-specified time intervals. The author used simulation to measure the performance of their solution. They assumed one node to be malicious and generated 5 pre-determined categories of attacks and measured the detection accuracy for each attack separately as well as the time spent on training, testing, and updating. They pre-set the number and categories of the feature vectors used for generating the spherical detectors.

In [74], the goal was to accurately identify abnormal measurements of medical WSNs. The authors assumed the existence of medical sensors that were attached to a patient's body, and responsible of measuring his vital biometric signs and sending them to a Local Processing Unit (LPU). The LPU would then measure the spatial correlation of the values it received to determine if an outlier exists. In such a case, the LPU generates an alarm as this outlier would typically indicate a health complication in the patient. Yet, the authors aimed to reduce the ratio of false alarms raised by the LPU by making sure the sensors do not send anomalous and/or faulty measures to the LPU. The authors assumed that the anomalous incidents were rather rare. Thus, the sensors carried out a lightweight distributed computation to identify anomalies locally, and communicate them to the LPU. The sensors utilized a forecasting technique in order to detect anomalies; hence, prior knowledge of the anomalous or normal data was not necessary. They also utilized the Exponentially Weighted Moving Average (EWMA) technique [75] for forecasting, and compared the expected measurement with the actual measurement to figure out the deviations from the expected behavior. The LPU applied chi-squared distance and Kernel Density Estimator (KDE) [76] to ignore uncorrelated data before raising a medical alarm. In their simulation, the authors used real medical dataset from the Physionet database[1]. The dataset contains almost 45000 records, and each record contains 4 attributes. They compared their approach with the Mahalanobis Distance (MD) [77], and found that the two methods performed similarly in terms of identifying anomalous values. They also injected 100 synthesized anomalies at different time instants and in different attributes, and measured the TP and FP rates.

In [29], the authors aimed at developing a self-learning anomaly detection solution that adapts to change in the network's traffic. Thus, they utilized a semi-supervised learning algorithm, where the detector was trained only on normal cases, and the knowledge about anomalous cases evolved dynamically. Consequently, they used the Discriminative Restricted Botlzman Machine (DRBM) neural networks [78], which combine strong generative modeling with classification accuracy in order to infer knowledge from incomplete training data. Typically, a DRBM comprises a network of stochastic neurons that

[1]`http://www.physionet.org/`, accessed on Nov. 4th, 2014

behave according to some energy model. The authors' hypothesis was that there exist deep similarities between normal cases that can be fully expressed using the DRBM machine so as to spot anomalous traffic with high accuracy. The authors conducted two sets of experiments; in one they acquired real traffic traces from a normal network host and an infected network host. In the second, they utilized the KDD'99 training set[2]. They measured the accuracy of the classifier as well as the free-energy associated with normal and anomalous traffic.

In [79], the authors aimed at selecting a subset of the data instances that is representative of the original data instances set. This data reduction would help design intelligent intrusion detection systems that are both effective and efficient. To achieve their goal, they introduced the notion of representativeness which measures the representative power of an instance within its class and with respect to the data instances set. An instance is considered representative when it is similar to many instances within its class and different from the other data instances within the other classes. In their work, they used a Euclidean distance metric to measure the similarity, and multiplied it by a factor that measures the similarity between the instance and all other instances within other classes. However, to reduce the cost associated with these measures, the authors used a centroid-based classification algorithm to measure similarities. To assess the effectiveness of their approach, they tested with several machine learning algorithms spanning: k-nearest neighbor, artificial neural networks, SVM, and liblinear classifiers, which are large-scale linear classifiers [80]. They compared the performance of these algorithms using the same dataset with and without representativeness and measured: TP and FP rates, precision, F-score, and accuracy. Their results showed that the reduced features resulted in detectors with performance comparable to the cases when no feature reduction was used. As for the data they used, it comprised labeled flow-based honeypot traces. As the majority of the traffic was anomalous, they incorporated benign traffic for the purpose of evaluation. They carried out extensive experimentation in order to figure out the proper values for the number of clusters as well as the number of representative instances.

In [81], the authors aimed at detecting anomalies with a high degree of accuracy albeit the conflicting network environment. Thus, they developed a novel combination of machine learning techniques for feature selection which consisted of k-means clustering, Naive Bayes classification, the Kruscal-Wallis test [82], and C4.5 decision trees. Their approach comprised:

1. An unsupervised stage in which k-means was applied in order to identify clustered features.
2. The feature selection step which utilized wrapper-based Naïve Bayes algorithm for relevant feature ranking and Kruscal-Wallis's test for significant feature ranking.
3. A supervised learning stage where the C4.5 decision tree algorithm was used to evaluate the significantly relevant subset of features selected in the selection step.

The authors found that statistically reduced yet relevant features can filter out noisy data associated with irrelevant features. Their work also showed that the speed and efficiency of anomaly detection improved, as reducing the amount of features helped decrease the amount of computation required. To assess their technique, they built a small office network and had it generate four types of Distributed Denial of Service (DDoS) attack to investigate whether or not their technique would identify the proper features for each type of DDoS attack. Firstly, they applied the k-means clustering algorithm which only gathered the malicious traffic into clusters. The number of clusters was comparable to the number of DDoS attack types. Then, they applied the Naïve-Bayesian classifier to rank the features within each cluster. Kruscal-Wallis method was applied afterwards to double check the significance of the previous steps. After the feature sets were identified, they got classified using the C4.5 algorithm. Then, the authors estimated the classification accuracy, FP and FN rates for the original features, the features

[2]`http://www.sigkdd.org/kdd-cup-1999-computer-network-intrusion-detection`, accessed: November 4th 2014

identified after the Naïve-Bayesian classifier, and the feature set identified after the Kruscal-Wallis test. Reducing the features did not significantly reduce the accuracy of the classification.
Another network anomaly detection solution that utilized machine learning is depicted in [83], where the authors took advantage of SVM, simulated annealing, and decision trees. SVM was used to select the best features, whereas simulated annealing and decision trees were used to obtain decision rules to help identify new attacks.

Other network anomaly detection solutions utilized statistical techniques. Here, we highlight a subset of the state-of-the-art statistical anomaly detection techniques in computer networks.
In [84], the authors aimed at developing a contextual anomaly detection technique that provides a scalable way to detect, classify, and interpret anomalies in sensor-based systems. This work outlined a contextual anomaly detection algorithm for streaming sensors of networks. This anomaly detection technique further enhances the parallel computing model, MapReduce in order to handle sensors' big data [85]. Additionally, the technique comprised a two-part detection scheme in order to detect point anomalies in real-time and then evaluate them using contextual clustering. The contextual clustering was performed using the profiles of the sensors that were used in similar contexts, to ensure real-time anomaly detection. In the part that detects point anomalies, the authors used a univariate Gaussian predictor that built a historical model of the data, and then predicted and compared new values based on the model. This part classifies data quickly at the expense of accuracy; thus, in the context-based anomaly detection part, the contextual anomaly detector defined the sensor profiles, and assigned each sensor to one of those profiles. It then evaluated the current value, pertaining to the sensor declared anomalous by the first part, against the sensor profile's average expected value. The sensor profiles were defined using a multivariate clustering algorithm which included sensors' multidimensional contextual metadata spanning location, time, and weather conditions. This part of the detector was learned off-line using MapReduce for computational efficiency. The authors utilized Powersmiths datasets to measure the performance of their proposed technique[3]. They also included artificial data streams during the testing phase to simulate real-time real-world scenarios. They experimentally selected 3 clusters, and counted the number of point-anomalies and contextual-anomalies that could be identified by their algorithm.

In [86], the authors aimed at developing an anomaly detector that adapts to analysis on irregular topologies, and can also analyze the traffic matrix in time as well as space domains. In this paper, the authors implemented Multi-Resolution Analysis (MRA) [87] on Traffic Matrix (TM) using Diffusion Wavelets (DW) [88]. As a typical TM for an $n-node$ network comprises N^2 entries, the MRA provides a sparse model of M entries where $M \ll N^2$. The DW implements the MRA on a given TM, and figures out the most important characteristics of the resulting sparse model. The authors utilized 84 samples from datasets of Abilene Network [89]. They introduced the concept of Contribution Ratio (CR), which is the ratio of the square of the volume V (either one of the 5^{th} level approximation coefficients generated by DW) to the energy of the traffic. The smoothness of the CR over a given time interval implied the absence of anomalous behavior. On the other hand, a change in the CR designated a sensor's disconnectivity, and thus implied the presence of an anomaly.

In [90], the authors aimed at developing a collective long-term anomaly detection algorithm for wireless sensor networks. Thus, they proposed a statistical data centric anomaly detection algorithm for wireless sensor networks that considered a collection of neighboring data segments as random variables. Each data segment comprised observations taken over a continuous time interval. Such a segment is said to be anomalous if it contains multiple contaminated measurements. In this work, anomalies were identified by measuring the minimum prediction variance of each data segment with respect to the rest. Hence, a given data-segment was labeled as anomalous if it was distinct in terms of its prediction variance. The authors assumed the WSN is split into clusters of adjacent sensors and developed a prediction variance

[3]"Powersmiths: Power for the Future," 2010, `http://ww2.powersmiths.com`, accessed on November 4th, 2014

detector accordingly. Each cluster had a Cluster Head (CH) that collected the data segments from the sensors within its cluster and considered them random variables. The CH predicted each variable, and obtained a set of prediction variances. After that, the CH constructed a statistical quantity based on a chi-squared distribution, and a confidence interval was established as the threshold. This quantity helped determine the sample covariance matrix among the variables which was used to help detect anomalies. The sensors did not transmit the measurements associated with their local data segments, but instead the compressed difference sequences, and sample standard deviations corresponding to their segments. This helped reduced the communication overhead by 80%, per the authors experiments, as the covariance matrix was built by taking advantage of the Spearman's rank correlation coefficient. The authors evaluated the proposed detector utilizing the measurements recorded by the IBRL sensor network which comprised 52 operating sensors[4]. They used a fixed-width clustering technique to split the network into 7 clusters. And all their experiments utilized one of the clusters that contained 11 nodes. They measured FP rate, the detection accuracy, and the average saving rate (i.e. communication cost) for 20 and 40 data segments. They also injected anomalies artificially in their experiments.

In [91], another collective anomaly detection solution was proposed, where a quickest change designates a sequence of observations whose Probability Density Function (PDF) might change at an unknown time. The goal of this paper is to detect the presence of such a change with a minimum delay under certain false alarm constraints. The authors assumed that the behavior of the network comprised a sequence of observations whose probability density function changed at unknown times. In their work, the authors assumed the network to be a wireless sensor network that is split into three groups, each of which applies the CUmulative SUM (CUSUM) strategy [92]. Each group that detected an anomaly sent an alarm to the fusion center. Upon receiving two alarms at least, the fusion center fired a global alarm to announce the detection of an anomaly. The sensors used two techniques for collecting observations within their own groups; the continuous-time Brownian motion and the discrete time model. The continuous-time Brownian motion assumes that changes in the behavior can happen simultaneously, while in the discrete model changes happen independently.

In [93], the authors aimed at developing a decentralized anomaly detection solution for wireless sensor networks in the presence of one or more classes of misbehaving nodes. The detector was assumed to operate in a wireless sensor network where a centralized fusion center exists. The fusion center collected feedback from the sensor, and for each anomaly it detected it decided the class to which the anomaly belonged. The authors assumed that each class of anomalies can be characterized using a decision rule (an operating point). Then, they reformulated this assumption into a maximum-likehood estimation problem with latent variables. After that, they applied the Expectation Maximization (EM) algorithm, which is an iterative algorithm for solving the maximum likelihood estimation problem. To assess the performance of their detection technique, the authors measured the discriminability which designates the rate of correctly classified cases, and the reliability which measures the distance between the estimated and actual operating points. They compared their technique with the with the Reputation-Based Classifier (RBC) algorithm [94], and showed that their approach outperformed the RBC algorithm in the metrics they measured.

Some recent network anomaly detection solutions used a hybrid approach that mixed different detection techniques. For example, the work in [95] aimed at developing a real-time anomaly detection solution for computer networks. In this work, the author added time to the features in order to accomplish anomaly detection in real-time. They also compared the performance of three well-known detection algorithms: multivariate normal distribution, k-nearest neighbor, and one-class SVM. To achieve their goal, the author generated a sequence of data points at regular time intervals, and then constructed a time series for each interval. After that, they created a feature vector for each time interval and mapped it to

[4]Intel lab data, `http://db.csail.mit.edu/labdata/labdata.html`. [Online]. accessed on November 4th, 2014

a single data point in the corresponding feature space. To evaluate the technique, they combined normal traffic from a University local network with 5 types of anomalous traffic from the Lincoln Laboratory at the Massachusetts Institute of Technology [96]. They measured precision, recall, and F-score for two sets of experiments; one with row features, and the other with the time-enhanced features for which they used a discrete wavelet transformation. Additionally, the features for each anomaly detection technique were selected manually. This work's results suggested that the proposed technique produced fine performance for several types of attacks.

Also, in [97], the goal was to compare and contrast the advantages and disadvantages of various categories of machine learning techniques for anomaly detection in computer networks. The authors did compare two statistical approaches to anomaly detection based on the Statistical Hypothesis Testing (SHT) with two methods based on 1-class SVM (i.e. one-class SVM) algorithm and one clustering technique. One of the SVM methods was a per-flow detector, while the other was window-based as it considered a sequence of flows within one time window and tried to detect anomalies within each window. The clustering technique was based on the Adaptive Resonance Theory (ART) [98] which splits the network into clusters based on the unique features of its flows. The ART detector was a per-flow detector, while the two SHT detectors were window-based. The authors did synthesize two types of traffic; flow-based and packet-based, using two traffic generating software tools they developed. The experiments showed that flow-based detectors yielded better resolution, but suffered from higher FP ratios and instability. On the other hand, the window-based techniques yielded lower FP ratios, and were more stable, yet exhibited lower resolution. The window-based 1-class SVM detector offered parameter tradeoffs between resolution and stability.

All the preceding recent anomaly detection solutions are behavioral; however, some solutions mixed the signature and behavioral approaches. One example is depicted in [62], where the goal was to utilize misuse detection in order to help select features and, hence, improve the performance of the solution. Thus, the authors proposed combining misuse detection with anomaly detection in a framework that comprised two phases; the first was the misuse phase which utilized the random forest algorithm to produce feature importance. In the cases where the misuse detection phase failed at identifying a potential threat, that threat's feature importance values got transferred to the anomaly detection phase (i.e. the second phase) which utilized weighted k-means algorithm to figure out if that potential threat was an anomaly. In their experiments, the authors injected anomalies into four datasets they complied based on the KDD'99 dataset, and measured the detection and the FP rates. They also calculated the importance of the features they selected. Their results showed that their proposed technique outperformed the classical machine learning techniques they compared with (i.e. random forest, SVM, cluster-based estimation, and k-nearest neighbor algorithms).

A recent collective anomaly detection solution that exploited the concept of spectral analysis is depicted in [99]. The authors aimed at addressing the special cases of collective anomalies in which anomalies appear locally in both time and space. They developed a new unsupervised spectral anomaly detection method that utilizes graph-based filtering framework. They also assumed that anomalous data instances were far less frequent than normal data instances. Their proposed technique is distributed in the sense that it splits a WSN into clusters, and assigns each cluster a cluster head to detect the anomalies within that cluster. In their work, they formulated the spectral decomposition using both graph-based filtering and PCA, and unified both formulations. Moreover, they showed that PCA is a special case of spectral decomposition with graph-based filtering. Their approach comprised three steps: graph construction, finding cut-off frequency of graph-based filters, and thresholding anomaly scores. In the graph construction step, an undirected graph that corresponds to the WSN was generated using the Euclidean distance between the sensors. In the next step, spectral decomposition was accomplished using graph-based filtering where normal and anomaly subspaces got separated for unsupervised detection. In the last step, a threshold score for detecting anomalies was selected. The authors defined their own scoring using projections on both normal and anomaly spaces. They also depicted the performance of their proposed graph-based filtering approaches by benchmarking against PCA-based and clustering methods in terms of ROCs and their AUC. Using the autoregressive model they generated a WSN of 100 nodes, and collective anomalies were generated using a highly varying autoregressive model at dedicated time instances. The LEACH protocol was used to split the network into 6 clusters and to select the cluster heads [100].

Other recent solutions for the problem of anomaly detection in networks use the information theoretic approach. For example, in the work depicted in [101], an active hypothesis testing technique was utilized to help solve the problem of quickest detection of a single anomalous node among a finite set of nodes. At specific time intervals, a subset of the network's nodes was observed and the observations of each node followed either one of two different distributions based on whether the node was normal or anomalous. The authors extended their solution so that they could handle multiple anomalous nodes under certain conditions. In another solution [54], the authors introduced the "typical day profile" technique and applied it to the change-point detection theory, which aimed at identifying sudden changes within a given time interval, to help identify anomalies in aggregate network traffic. The authors also analyzed and contrasted some change point detection methods in order to detect anomalies in a dataset of 24-hour worth of network traffic within a university campus.

Some recent solutions to the anomaly detection problem in computer networks have utilized the streaming technique. For example, the work depicted in [102], introduced the LD-Sketch, which is a data structure intended for accurate and scalable traffic anomaly detection. The LD-Sketch combines counter-based and sketch-based techniques, and operates in two phases. In the first of which, local detection was performed. Then, distributed detection was applied to reduce false positives by aggregating multiple detection results. The authors compared their solution with some state-of-the-art streaming techniques including [103, 67, 104], and showed that their solution achieved better accuracy. Another solution that is based on the streaming approach is depicted in [104], where the authors proposed the fast sketch data structure which can aggregate packets into a small number of flows, to detect traffic anomalies with small space and time. Besides, their sketch combined both the combinatorial group testing, and the quotient technique to identify anomalies to help achieve faster anomaly detection.

Table 1 summarizes the anomaly detection solutions described in this section. The approach column designates the detection technique utilized in the solution. Network Centric (NC) Vs. Data Centric (DC) specifies whether the solution addressed anomalies in terms of the network packet headers, or in terms of packets payload. The Scalable? column designated whether scaling for larger networks is taken into account in the solution or not. In this table, Inf. Theoretic stands for the information theoretic approach.

3.2 Application-level Anomaly Detection Solutions

In the context of smartphone networks, security has been a crucial issue. Thus, most of the anomaly detection solutions in this context address malware detection in particular, although several such solutions can potentially handle other anomalous behaviors should they be exposed to their corresponding data instances. A vast majority of malware detection techniques are behavior-based. We first explore the literature on behavior-based static analysis malware detection techniques for mobile devices.

In [105], static analysis was used to gather as many features as possible. The proposed system, DREBIN, is a lightweight system that can run on the device itself. The authors statistically inspected a given Android application and extracted different feature sets from the manifest and .dex files. The extracted features sets were then mapped to a joint vector space, such that patterns and combinations of features could be analyzed geometrically. Then, SVM was applied to help separate malware behavior from normal behavior. Upon detection a malicious application, the features pertaining to that application were reported back to the user. In this work, the feature sets included hardware components, requested

Table 1: Summary of recent anomaly detection solutions at the network level.

1st Author, Year and Ref.	Approach	NC vs. DC	Scalable?	on/off-line
AbuAitah, 2014 [71]	Machine learning	DC	No	Online
Barani, 2014 [73]	Machine learning	NC	Yes	Online
Salem, 2014 [74]	Machine learning	DC	No	Online
Fiore, 2013 [29]	Machine learning	NC	No	Offline
Guo, 2013 [79]	Machine learning	NC	No	Online
Louvieris, 2013 [81]	Machine learning	NC	No	Offline
Lin, 2012 [83]	Machine learning	NC	No	Offline
Hayes, 2014 [84]	Statistical	DC	Yes	Online
Sun, 2014 [86]	Statistical	NC	No	Offline
Xie, 2014 [90]	Statistical	DC	Yes	Online
Bayraktar, 2014 [91]	Statistical	DC	No	Online
Soltanmohammadi, 2013 [93]	Statistical	DC	No	Online
Limthong, 2013 [95]	Mixed	NC	No	Online
Wang, 2013 [97]	Mixed	NC	No	Offline
Egilmez, 2014 [99]	Spectral	DC	Yes	Online
Cohen, 2014 [101]	Inf. Theoretic	DC	No	Online
Cuadra-Sanchez, 2014 [54]	Inf. Theoretic	NC	No	Offline
Huang, 2014 [102]	Streaming	NC	Yes	Online
Liu, 2012 [104]	Streaming	NC	No	Online

permissions, application components, filtered intents, restricted API calls, suspicious API calls, and MAC addresses. The authors evaluated their detection solution using 123453 applications and 5560 malware samples. They performed three sets of experiments, in the first they measured the detection performance of DREBIN and compared it to sKirin [106], RCP [107], and the approach by Peng et al. [108], as well as some Anti-Virus engines. They measured the TP and FP rates, and found that their approach outperformed the other static-analysis techniques they compared with, and was comparable to the anti-virus engines. In the second set, they investigated the explainability for 4 malware families. For each sample of those families, they identified the features with the highest contribution to the classification decision, and then averaged the results for all the instances of a family. In the third set, they ran an already learned DREBIN instance on several smartphones, and found that the execution time required to analyze an application was 10 seconds on average.

In [109], the authors also utilized static analysis for malware detection. They obtained permission combinations that are frequently requested by malware applications, and implemented a rule-based malware detection scheme which they called Droid Detective. Their approach comprised decomposing the application into several files and extracting the manifest file in a readable format. Then, they fed the manifest file to their permission mapper, which kept track of the permission combinations seen so far along with their appearance frequencies. Next, the permission maps (i.e. permission combinations together with their frequencies) were fed to the rule sets selection process. The rule sets selection process was responsible for extracting the permission combinations that were requested more often by malware than by benign Apps, according to an equation developed by the authors. In their experiments, the authors measured the TP, TN, FP, and FN rates utilizing 1260 malware samples published by NCSU researchers [110], together with 741 benign applications collected from the official Google Android Market. Their experiments revealed that their permission-combination approach managed to detect the SMS related malware efficiently with low FP rate.

Another static analysis solution is depicted in [34], were the authors aimed at utilizing statistical approach to solve the problem of malware detection using static analysis. They used a probabilistic discriminative model based on regularized logistic regression for Android malware detection. Also, the authors decompiled applications' source codes and extracted Android API calls as features (i.e. source code features), in addition to application permissions' features. The authors explored issues in feature granularity, feature representation, and feature selection and regularization. Moreover, they utilized information gain and chi-squared techniques for feature selection. In their experiments, the authors used datasets collected from different sources. They measured the precision, recall, and the AUC for their proposed system. Moreover, they compared their system with Naive Bayes with Informative Priors (PNB), Hierarchical Mixture of Naive Bayes (HMNB), and 2-PNB generative models based on permissions' features in one set of experiments [108]. In the other set, they compared their system with the same 3 generative models, as well as k-nearest neighbor, SVM, and decision trees, based on source code features. In the third set, they combined the source code features and application permission features in different ways and measured the precision and recall of their proposed solution. The authors showed that their system outperformed the other state-of-the-art methods they compared with based on static analysis.

In [111], the authors aimed at developing a rapid static analysis malware detection solution. They combined feature-based applications permission, broadcast receivers, and the presence of embedded Android applications and native code. They utilized random decision trees that built rules comprising 2 or 3 features each. To assess the performance of their system, they obtained a set of known malicious applications and some benign applications drawn randomly from a set of 500 applications. They measured TP and FP ratios, illustrated that their system performed comparably to leading antivirus engines.

Other static analysis solutions that utilized machine learning techniques are depicted in [112] and [113]. In [112], reverse engineering of the application's executable files was used to obtain the feature set using a Java-based profiling tool developed by the authors. Also, the authors utilized a Bayesian classifier for their learning algorithm. In [113], one-class SVM was utilized to help identify malware behavior while using the open-source Androguard tool[5] for feature extraction. Both solutions are offline and off-device.

A wider set of recent malware detection solutions for mobile devices has adopted the behavior-based dynamic analysis approach. Here, we highlight a subset of thee recent techniques.
In [114], the authors aimed at combining the best features from host-based and cloud-based anomaly detection techniques. Thus, they developed a dynamic hybrid mobile IDS framework with both host-based and cloud-based capabilities. Their framework is intended for iOS mobile devices, but can be extended to accommodate Android devices as well. The framework incorporated four diverse detection mechanisms, and anomalous behavior was determined based on their combined outcomes. These mechanisms comprised: SMS profiling, monitoring of applications behavior via iDMA [115], a touch logger profiler (iTL) [116], and a keystroke-based authentication system. Upon receiving an alert regarding a suspicious event, the framework would determine whether to run the detection algorithm on the cloud or the host itself, and then will inform either one to accomplish the detection task. Upon detecting an anomalous application, it decides what the actions to be taken and adds the application's relevant information to its own behavior profile database. To evaluate their framework, the authors collected traces from real users' profiles, and utilized decision trees for their learning algorithms. They ran experiments to measure the detection accuracy as well as the CPU, memory, battery, and timeliness (i.e. the time to detect a malicious application).

[5]Androguard. `https://code.google.com/p/androguard/`, accessed on November 4th, 2014

In [117], the authors aimed at developing a dynamic system for detecting a mobile application's meaningful deviations from the normal behavior based on its network traffic. They targeted republished popular applications injected with malware or malicious code as well as malware with self-updating capabilities. As each application should have a model that describes its traffic patterns, they aimed at learning such models using a semi-supervised machine learning algorithm that learns the behavior of normal applications. Thus, when such an algorithm is presented with a new application, it would measure its deviation from normal behavior. If the measured deviation exceeds a pre-defined threshold, the application will be considered a malware. Their proposed system consisted of 6 main components:

1. A Graphical User Interface (GUI) that communicated with the user and displayed alerts.
2. An alerts handler that generated alerts and processed users' responses to them.
3. Feature extraction which measured specific features at specific time intervals.
4. Feature aggregation which computed defined aggregations over all extracted measurements within a specified time interval.
5. A local learner that comprised local models that represented the applications' traffic patterns pertaining to the user.
6. An anomaly detector which performed online analysis of the applications' network behavior, and detected the deviations from normal behavior.

The learning algorithm, in this work, was based on the cross-feature analysis approach which transforms a semi-supervised learning problem to a collection of supervised learning problems for which well-established solutions exist [118]. Moreover, detection and learning processes took place locally on the device itself. The authors conducted several sets of experiments utilizing the applications' traffic pertaining to the devices of 8 volunteer users over a defined period of time. They also used 10 self-written and 5 real malware applications for testing. Their experiments revealed that modeling an application's traffic using only application-level features is possible, also they deduced 7 discriminating network-relevant application-level features. For the cross-feature analysis they used C4.5 decision trees for categorical features and several machine learning techniques for the numerical features, including regression-based classifiers, Decision Table [119], SVM, and Decision/Regression Tree (REPTree)[6]. The authors measured the TP and the FP rates, the accuracy rate, and detection time. Their results showed the REPTree and Decision Table to be most successful in terms of the metrics they measured. The authors argued the efficiency of their solution in terms of memory and CPU overhead, which would encourage its deployment on the devices directly. However, these overheads did not cover the feature extraction and aggregation components.

In [120], the authors aimed at achieving rapid dynamic large-scale detection of mobile malware using machine learning algorithms. They presented their STREAM framework which is a distributed application profiler that comprises a master server that distributes profiling jobs to worker nodes. The worker nodes, in their turn, distribute the jobs among the devices and/or emulators to run the detection algorithms in parallel. The STREAM master then harvests the feature vectors from the devices and emulators and handles classifier training and evaluation. In their framework, they evaluated random forests, multilayer perceptron networks, Naive Bayesian networks, logistic regression, and J48 decision trees. In

[6]`http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html`, accessed on November 4th 2014

their experiments, they utilized a database of known malwares in order to evaluate their framework. Besides, they utilized Android's MonkeyRunner[7] user input tool to explore the application's behavior. The authors measured the TP rate, the FP rate, and the detection time, as well as the percentage of correctly classified instances. Their results showed that the Naive Bayesian networks performed the best, while the logistic regression algorithm performed the least in terms of the metrics they measured. Moreover, they trained their algorithms on 1330 malware applications and 408 benign applications.

The authors in [35] proposed a set of features to detect mobile malware in mobile devices using machine learning techniques dynamically. In their work, the authors proposed an Android malware detection system that consisted of a data processing step which comprised deploying an agent on the mobile device. The agent monitored the behavior of the applications executing in the device and recorded some measurements in a pre-defined format. The solution also consisted of a malware detection component which created a machine learning model for each application. This component applied 4 types of classifiers: Naive-Bayesian networks, random forest, logistic regression, and SVM, and measured the detection performance for each classifier. The features they identified spanned: network related features, SMS related features, CPU related features, power consumption features, process relevant features, memory relevant features. In their experiments, the authors measured the performance of the classifiers they experimented with in terms of precision, recall, F-score, and ROC AUC. They utilized 30 normal applications and 5 malware applications, and utilized the information gain technique to identify the best performing features among the 32 features they considered initially. According to their experiment, the top 10 performing features addressed memory, CPU, network, and SMS aspects of the application. Their experiments showed that the decision trees outperformed the remaining machine learning algorithms in terms of TP and FP rates.
In [121], the authors aimed at introducing the service-oriented architecture concepts to malware detection solutions. Thus, they proposed a client-server architecture called SmartMal, where clients continuously extracted various features and transferred them to the server. The server side comprised distributed servers that run state-of-the-art anomaly detection techniques in parallel, in order to analyze clients' feature vectors. Besides, their architecture utilized information fusion in order to concatenate the servers' results. For anomaly detection, they proposed a model-based approach where an Analytical Hierarchy Process (AHP) [122] was used to construct a 3-layer model in order to divide the complex strategic decision problem into different subjects aiming at several targets. For each of these subjects, a fuzzy quantitative approach was utilized to estimate the weights of the model. To evaluate the performance of SmartMal, the authors built a prototype to profile the behavior of normal applications. They used 3 smartphones on which 32 benign applications ran and sampled their status every 30 seconds via a background engine. Then, they measured the CPU/memory utilization, the battery consumption, and the network traffic profile for those applications. To measure the performance of their architecture, they utilized three smartphones, to simulate and detect a DoS attack. The authors also compiled feature vectors for a defined amount of time using the measurement recorded by the devices, and measured the TP rate.

Other dynamic analysis solutions for malware detection were proposed in [36, 123, 124, 37]. In [36], the authors aimed at determining the best features selection technique with the best machine learning algorithm. They experimented with 5 machine learning algorithms, spanning Naive Bayesian, k-nearest neighbor, J48 decision trees, multilayer perceptron, and random forest. They utilized the chi-squared and information gain methods to help identify the best features per the system calls they captured in their experiments alongside different features from [125, 126, 127]. In [123], the random forest classifier was applied to the data observed from 1330 malicious applications and 407 benign applications. In [124], Naive Bayesian and SVM learning algorithms were utilized on malware and benign samples. The techniques in [36, 123, 124] are all offline and off-device. In [37], malicious application behavior was

[7] `http://developer.android.com/tools/help/monkeyrunner_concepts.html`, accessed on November 4th, 2014

detected online and on-the-device. For the monitored applications under execution, several metrics were measured and submitted to different analysis units (i.e. processes). The feedback from each process was weighted to help reduce the probabilities of false alarms. Upon discovering a malicious application, the system would submit an alert to the user alongside a set of suitable actions to mitigate the malicious behavior. The authors used chi-squared, information gain, and Fisher-score methods to help select the best features for their learning algorithms. They experimented with several learning and statistical algorithms spanning decision trees, Naive Bayesian networks, k-means, histogram, and logistic regression.

Some other recent techniques for malware anomaly detection adopted the signature-based rather than the behavior-based approach. In such solutions, a pre-defined set of malware behaviors are formulated and hardwired into the solution itself. In [128], for example, the authors aimed at performing anomaly detection for malware in mobile phones using dynamic analysis at a large scale and in real-time. They presented Andlantis, a scalable dynamic analysis system capable of processing over 3000 Android applications per hour. Andlantis architecture comprised:

1. Kane, which is a commodity cluster for malware detection analysis constituting 520 nodes of commodity PCs.

2. FARM, which was responsible for malware collection and aggregation. Forensic Analysis Repository for Malware (FARM) stored known-bad, known-benign, and unknown applications and results from automated analysis tools. Upon deploying Andlantis, the automated analysis tool of the administrator choice can be utilized.

3. Andlantis agent, which was responsible for job control and scheduled where a sample will be analyzed and examined for malicious behavior. The main criterion for selecting a node was the availability of its resources.

4. Minigma, which is a large scale Virtual Machine (VM) management tool to control the Kane devices.

5. Dynamic analysis or application interaction, which carried out the detection task.

The nodes that examined the applications in order to analyze their runtime behavior utilized an emulator with a GUI that was controlled via an application based on the MonkeyRunner and AndroidViewClient frameworks[8]. This helped perform easy and customizable UI exploration while the application was analyzed. To evaluate the performance of their proposed system, the authors collected forensic data and utilized 1261 malware samples. They measured the execution time and showed its decompositions into computation time and communication time.

Another signature-based malware detection solution is presented in [129], where the authors proposed the dynamic analysis tool DroidTrace. This tool is equipped with forward execution capability (i.e. physical modification), and utilizes ptrace, a Linux system call, to monitor selected system calls. In their solution, the authors focused on detecting dynamic loading which allows an application to load a dynamic library into the memory while it is executing. Their forward execution capability helps trigger automatically almost all the dynamic loading behaviors in the application. DroidTrace is online and can operate on the device itself. In [130], DriodScope, the offline and off-device dynamic analysis tool, was proposed to achieve malware detection via operating on top of the emulator while reconstructing some Android OS components as well as Java-level semantics simultaneously. This tool monitors an application while it executes and collects its native and Dalvik[9] instructions traces along with API-level activities. It also accounts for data leakage using taints, where some data items get specific labels, and

[8]`https://github.com/dtmilano/AndroidViewClient`, accessed on November 4th, 2014
[9]`https://source.android.com/devices/tech/dalvik/`, accessed on November 4th, 2014

are monitored as they circulate within the system. When these taints get communicated out of the device, the detection solution gets notified.
The solution proposed in [131] mixed signature-based with heuristic-based approaches for malware detection in mobile devices. The authors proposed an online off-device solution called DroidRanger, where they started with a permission-based static analysis to detect samples of malware. Then, applied a heuristic technique that checked for two types of malicious behaviors: dynamic loading from untrusted websites, and loading native code in unusual way.

A brief summary of the solutions depicted in this subsection is depicted in Table 2. The summary highlights the major characteristics of these solutions. In this table, *Stat./Dyn.* stands for static/dynamic, while *Appr.* stands for approach. *ML* means machine learning, *Stat.* means statistical, and *Algo.* means algorithmic. *Scal.?* stands for Scalable?, and *Sign./Behav.* stands for Signature-based/Behavioral.

Table 2: Summary of recent anomaly detection solutions at the application level.

1st Author, Year and Ref.	Stat./Dyn.	on/off-device	Appr.	Scal.?	on/off-line	Sign./Behav.
Damopoulos, 2014 [114]	Dynamic	Mixed	ML	Yes	Online	Behavioral
Shabtai, 2014 [117]	Dynamic	On-device	ML	Yes	Online	Behavioral
Amos, 2013 [120]	Dynamic	Off-device	ML	Yes	Online	Behavioral
Ham, 2013 [35]	Dynamic	Off-device	ML	No	Offline	Behavioral
Wang, 2013 [121]	Dynamic	Off-device	Stat.	Yes	Online	Behavioral
Mas'ud, 2014 [36]	Dynamic	Off-device	ML	No	Offline	Behavioral
Alam, 2013 [123]	Dynamic	Off-device	ML	No	Offline	Behavioral
Wei, 2013 [124]	Dynamic	Off-device	ML	No	Mixed	Behavioral
Shabtai, 2012 [37]	Dynamic	Off-device	ML	No	Online	Behavioral
Yan, 2012 [130]	Dynamic	Off-device	–	No	Offline	Signature
Zheng, 2014 [129]	Dynamic	On-device	–	Yes	Online	Signature
Zhou, 2012 [131]	Dynamic	Off-device	Algo.	No	Online	Mixed
Arp, 2014 [105]	Static	On-device	ML	Yes	Online	Behavioral
Liang, 2014 [109]	Static	Off-device	ML	No	Offline	Behavioral
Cen, 2014 [34]	Static	Off-device	Stat.	No	Offline	Behavioral
Glodek, 2013 [111]	Static	Off-device	ML	No	Offline	Behavioral
Yerima, 2013 [112]	Static	Off-device	ML	No	Offline	Behavioral
Sahs, 2012 [113]	Static	Off-device	ML	No	Offline	Behavioral

4 Summary of subtleties

In this section, we highlight the major subtleties and drawbacks in the various anomaly detection solutions we investigated. Emphasizing these issues should help in designing better anomaly detection solutions in the future.

- Several anomaly detection techniques assume that the vast majority of network traffic is normal, and that anomalous traffic is much less frequent. However, this assumption does not always hold, on the contrary, under some network-wide failures and/or attacks the opposite is true.

- Several anomaly detection solutions assume that the initial measurements recorded by a given node are normal and based on that, the solution decides whether or not some other node is anomalous. But, it is not guaranteed that a node's initial measurements are normal by default.
- Several distributed anomaly detection solutions assume that the results and/or measurements communicated by a given node in the network are genuine, regardless of being normal or anomalous. However, a malicious node can, for instance, choose to deliberately send falsified measurements other than the ones it actually calculated. For example, it may choose to report normal measurements to confuse the anomaly detector.
- Several distributed anomaly detection solutions fail to address the problem of messages being compromised during transmission. Compromising messages would typically confuse the anomaly detector as well.
- The anomaly detection solutions that deployed a centralized fusion center to collect node measurements and identify anomalies nodes accordingly overlook the case when the fusion center itself is anomalous and/or when it fails. So, such solutions comprise a single-point of failure condition.
- In some anomaly detection solutions, a node needs to identify whether or not its own traffic is normal. However, it is impractical to assume than even a normal node can always distinguish normal from anomalous traffic.
- The feature selection method in many anomaly detection solutions is either not automated, or is based on a given dataset. However, developing automatic feature selection that would perform as desired regardless of the dataset is to be realized yet.
- Some machine learning solutions require considerable amounts of training time which renders them impractical.
- Several solutions oriented to sensor and smartphone networks lack thorough experimentation to support their claims on saving the devices' resources and energy.
- In many solutions, datasets of normal and/or anomalous instances are synthesized in order to assess the performance of the solutions. However, this is not always sound due to the following reasons:
  - Datasets are artificially generated according to some pre-defined models which are not necessarily realistic, and would affect the soundness of the results.
  - Datasets are compiled by mixing instances from different sources, which generates another kind of unrealistic datasets that would also affect the soundness of the results.
  - In some cases, anomalous traffic was injected into normal traffic within a dataset at some pre-defined intervals. Again, this results in an artificial dataset which could produce misleading detection results.
  - Some of the realistic datasets are obsolete, and do not necessarily reflect the nature of anomalies that currently exist in computer networks.
- In the solutions that utilize the concept of clusters, the number of clusters is manually picked and mostly depends on the fact that the experiments address a pre-defined set of anomalous behaviors. This would limit the practicality of such solutions, as in reality determining the number of anomalous behavior classes is rather infeasible; furthermore, predefining the set of anomalous behaviors may lead to the inability to detect completely new anomalies in a way that is similar to the problem observed in signature-based detectors.

- Some solutions assume that there exists one class of anomalous behavior to be detected. However, practically, more than one anomalous class exist, not to mention unseen classes of normal behavior.
- Some solutions claim being timeless and responsive, but few provide solid experimentation results to support their claims.
- Most of anomaly detection solutions address point anomalies, and very few of them address contextual or collective anomalies.
- Several solutions fail to address network dynamics, especially in sensor and smartphone networks, where nodes may join and leave the network freely.
- Several detection solutions can identify certain classes of anomalous behavior, but cannot track it and locate the node(s) that exhibited that behavior which is rather crucial for the practicality of these solutions.

5 Conclusions

In this review, we discussed the state-of-the-art solutions of the anomaly detection problem in computer networks. We identified two levels for handling this problem; the network level and the application level. The network-level detection, on one hand, analyzes the headers and/or the payloads of the messages exchanged in the network, while application-level detection analyzes the application specification and/or examines its behavior during runtime.
Tackling anomaly detection in computer networks has been handled using several techniques belonging to different disciplines. One approach to solve anomaly detection is using either parametrized or non-parametrized statistical techniques to model the network and/or devices behavior and measure the deviation of anomalous behaviors from normal ones. Alternatively, machine learning techniques can be used to learn the normal and/or abnormal behaviors, and then try to classify or cluster unseen behaviors accordingly. In addition, information theoretic as well as spectral techniques provide different perspectives to help measure how an anomalous behavior differs from normal behaviors. Streaming techniques can be also used in the context of anomaly detection, where instead of sampling behavior information at regular intervals, the whole behavior information gets summarized into sketches to allow detecting anomalous behaviors.

Generally speaking, future anomaly detection solutions need to efficiently and effectively handle the shortcomings that many current solutions suffer from. For example, albeit achieving high detection rates, several anomaly detection solutions suffer from relatively high false positive rates which renders them impractical. Additionally, the majority of recent anomaly detection solutions utilize machine learning techniques, although such techniques merely identify correlations rather than causation. Thus, such detection solutions would provide little insights when the mitigation and/or justification of the anomalous behavior is sought. Also, as noticed in the studies depicted in this review, most anomaly detection solutions are intended for point anomalies, and fewer solutions are either contextual or collective. However, effective anomaly detection would naturally go beyond point anomalies to tackle anomalies within contexts and collectively anomalous behaviors. Moreover, many anomaly detection solutions overlook issues pertaining to the dynamics of the networks they investigate, especially for wireless and mobile networks, while others assume that anomalous behaviors or even normal behaviors do follow specific models that are known a priori.
Additionally, many anomaly detection solutions for mobile networks lack a rigorous assessment of the amounts of energy they consume either for local devices' computation, or for communicating devices'

statistical information over the network. An additional vital drawback in many anomaly detection solutions is that, although they assume networks and/or malware attacks are the reason behind the anomalous behaviors, such solutions do not try to secure the communication paths over which detection information is being exchanged. Also, they do not address the situations in which nodes themselves could be sending forged messages to mislead the anomaly detection solution.

References

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey," *ArXiv e-prints*, May 2013.

[2] L. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, November 2014.

[3] Z. Bi, L. D. Xu, and C. Wang, "Internet of things for enterprise systems of modern manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1537–1546, May 2014.

[4] S. Li, L. Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, pp. 1–17, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10796-014-9492-7

[5] J. Fidalgo and J. Lopes, "Load forecasting performance enhancement when facing anomalous events," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 408–415, February 2005.

[6] C. Brighenti and M. Sanz-Bobi, "Auto-regressive processes explained by self-organized maps. application to the detection of abnormal behavior in industrial processes," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2078–2090, December 2011.

[7] T. Auld, A. Moore, and S. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 223–239, January 2007.

[8] D. Wijayasekara, O. Linda, M. Manic, and C. Rieger, "Fn-dfe: Fuzzy-neural data fusion engine for enhanced resilient state-awareness of hybrid energy systems," *IEEE Transactions on Cybernetics*, vol. 44, no. 11, pp. 2065–2075, November 2014.

[9] N. Gobbo, A. Merlo, and M. Migliardi, "A denial of service attack to gsm networks via attach procedure," in *Security Engineering and Intelligence Informatics - Proc. of the CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, LNCS*, vol. 8128. Springer-Verlag, September 2013, pp. 361–376.

[10] W. Liping, "An anomaly detection based on local wave decomposition and clustering," in *Proc. of the 2010 International Conference on Information Networking and Automation (ICINA'10), Kunming, China*, vol. 2, October 2010, pp. V2–390–V2–393.

[11] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Adaptable n-gram classification model for data leakage prevention," in *Proc. of the 7th International Conference on Signal Processing and Communication Systems (ICSPCS'13), Gold Coast, Australia*, December 2013, pp. 1–8.

[12] E. Hormozi, M. Akbari, M. Javan, and H. Hormozi, "Performance evaluation of a fraud detection system based artificial immune system on the cloud," in *Proc. of the 8th International Conference on Computer Science Education (ICCSE'13), Phuket, Thailand*. IRCSIT Press, April 2013, pp. 819–823.

[13] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1460–1470, July 2013.

[14] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, and B. Furht, "Sensor fault and patient anomaly detection and classification in medical wireless sensor networks," in *Proc. of the 2013 IEEE International Conference on Communications (ICC'13), Budapest, Hungary*. IEEE, June 2013, pp. 4373–4378.

[15] A. Armando, A. Merlo, M. Migliardi, and L. Verderame, "Breaking and fixing the android launching flow," *Computers & Security*, vol. 39, pp. 104–115, November 2013. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2013.03.009

[16] Y. Wang, E. Ma, T. Chow, and K.-L. Tsui, "A two-step parametric method for failure prediction in hard disk drives," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 419–430, February 2014.

[17] S. Cheng, K. Tom, and M. Pecht, "Anomaly detection of polymer resettable circuit protection devices," *IEEE Transactions on Device and Materials Reliability*, vol. 12, no. 2, pp. 420–427, June 2012.

[18] H. Nguyen, Z. Shen, Y. Tan, and X. Gu, "Fchain: Toward black-box online fault localization for cloud systems," in *Proc. of the 33rd IEEE International Conference on Distributed Computing Systems (ICDCS'13), Philadelphia, USA*. IEEE, July 2013, pp. 21–30.

[19] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, July 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176

[20] P.-Y. Chen, S. Yang, and J. McCann, "Distributed real-time anomaly detection in networked industrial sensing systems," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, pp. 1–1, 2014.

[21] M. E. Shin, "Self-healing components in robust software architecture for concurrent and distributed systems," *Science of Computer Programming*, vol. 57, no. 1, pp. 27–44, July 2005. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2004.10.003

[22] J. B. Cabrera, C. Gutiérrez, and R. K. Mehra, "Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks," *Information Fusion*, vol. 9, no. 1, pp. 96–119, 2008, special Issue on Applications of Ensemble Methods. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253507000425

[23] M. Burgess, "Probabilistic anomaly detection in distributed computer networks," *Science of Computer Programming*, vol. 60, no. 1, pp. 1–26, March 2006. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2005.06.001

[24] H. Jeong, W. Hyun, J. Lim, and I. You, "Anomaly teletraffic intrusion detection systems on hadoop-based platforms: A survey of some problems and solutions," in *Proc. of the 15th International Conference on Network-Based Information Systems (NBiS'12), Melbourne, Australia*. IEEE, September 2012, pp. 766–770.

[25] V. Raychoudhury, J. Cao, M. Kumar, and D. Zhang, "Middleware for pervasive computing: A survey," *Pervasive & Mobile Computing*, vol. 9, no. 2, pp. 177–200, April 2013. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2012.08.006

[26] M. Xie, S. Han, B. Tian, and S. Parvin, "Anomaly detection in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1302–1325, 2011, advanced Topics in Cloud Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804511000580

[27] H. Nallaivarothayan, D. Ryan, S. Denman, S. Sridharan, and C. Fookes, "An evaluation of different features and learning models for anomalous event detection," in *Proc. of the 2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA'13), Hobart, Australia*, November 2013, pp. 1–8.

[28] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," *Computers & Security*, vol. 30, no. 6–7, pp. 353–375, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404811000691

[29] U. Fiorea, F. Palmierib, A. Castiglionec, and A. D. Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, December 2013.

[30] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, July 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[31] D. Difallah, P. Cudre-Mauroux, and S. McKenna, "Scalable anomaly detection for smart city infrastructure networks," *IEEE Internet Computing*, vol. 17, no. 6, pp. 39–47, November 2013.

[32] E. Anceaume, Y. Busnel, E. Le Merrer, R. Ludinard, J. Marchand, and B. Sericola, "Anomaly characterization in large scale networks," in *Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'14), Atlanta, USA*. IEEE, June 2014, pp. 68–79.

[33] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, First 2014.

[34] L. Cen, C. Gates, L. Si, and N. Li, "A probabilistic discriminative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, September 2014.

[35] H.-S. Ham and M.-J. Choi, "Analysis of android malware detection performance using machine learning classifiers," in *Proc. of the 2013 International Conference on ICT Convergence (ICTC'13), Jeju Island, Korea*, October 2013, pp. 490–495.

[36] M. Mas'ud, S. Sahib, M. Abdollah, S. Selamat, and R. Yusof, "Analysis of features selection and machine learning classifier in android malware detection," in *Proc. of the 2014 International Conference on Information Science and Applications (ICISA'14), Seoul, Korea*, May 2014, pp. 1–5.

[37] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161–190, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10844-010-0148-x

[38] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[39] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill, Inc., 1997.

[40] A. K. Sharma, *TextBook Of Chi-Test And Experimental Designs*, 1st ed. Publishing House, 2005.

[41] J. L. Crowley, J. H. Piater, M. Vincze, and L. Paletta, Eds., *Proc. of the 3rd International Conference on Computer Vision Systems (ICVS'03), Graz, Austria.* Springer-Verlag, April 2003.

[42] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, March 2003.

[43] H. Al Marakeby, M. Zaki, and S. Shaheen, "A generalized object detection system using automatic feature selection," in *Proc. of the 10th International Conference on Intelligent Systems Design and Applications (ISDA'10), Cairo, Egypt*, November 2010, pp. 839–844.

[44] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, *A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection*. SIAM, 2003, ch. 3, pp. 25–36. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611972733.3

[45] M. Migliardi and A. Merlo, "Improving energy efficiency in distributed intrusion detection systems," *Journal of High Speed Networks*, vol. 19, no. 3, pp. 251–264, 2013.

[46] S. Kumar, C. Rama Krishna, N. Aggarwal, R. Sehgal, and S. Chamotra, "Malicious data classification using structural information and behavioral specifications in executables," in *2014 Recent Advances in Engineering and Computational Sciences (RAECS'14)*, March 2014, pp. 1–6.

[47] A. Karnik, S. Goswami, and R. Guha, "Detecting obfuscated viruses using cosine similarity analysis," in *Proc. of the 1st Asia International Conference on Modelling Simulation (AMS '07), Pukhet, Thailand.* IEEE, March 2007, pp. 165–170.

[48] J. Kittler, W. Christmas, T. de Campos, D. Windridge, F. Yan, J. Illingworth, and M. Osman, "Domain anomaly detection in machine perception: A system architecture and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 845–859, May 2014.

[49] M. Thottan, G. Liu, and C. Ji, "Anomaly detection approaches for communication networks," in *Algorithms for Next Generation Networks*, ser. Computer Communications and Networks, G. Cormode and M. Thottan, Eds. Springer London, 2010, pp. 239–261. [Online]. Available: http://dx.doi.org/10.1007/978-1-84882-765-3_11

[50] W. Holmes Finch, J. E. Bolin, and K. Kelley, *Multilevel Modeling Using R*, 1st ed. CRC Press, 2014.

[51] R. A. Rutledge, *Just Enough SAS: A QuickStart Guide to SAS for Engineers*. SAS Publishing, 2009.

[52] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, June 2008. [Online]. Available: http://dx.doi.org/10.1214/009053607000000677

[53] J. Wang, *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. Springer Publishing Company, Incorporated, 2012.

[54] A. Cuadra-Sanchez, J. Aracil, and J. Ramos de Santiago, "Proposal of a new information-theory based technique and analysis of traffic anomaly detection," in *Proc. of the 2014 International Conference on Smart Communications in Network Technologies (SaCoNeT'14), Vilanova i la Geltru, Spain*, June 2014, pp. 1–6.

[55] L. C. Jain and N. T. Nguyen, *Knowledge Processing and Decision Making in Agent-Based Systems*, 1st ed.

Springer Publishing Company, Incorporated, 2009.

[56] A. Nigrin, *Neural Networks for Pattern Recognition*. MIT Press, 1993.

[57] M. Gardner and S. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” *Atmospheric Environment*, vol. 32, no. 14–15, pp. 2627–2636, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1352231097004470

[58] D. Heckerman, “A tutorial on learning with bayesian networks,” in *Innovations in Bayesian Networks*, ser. Studies in Computational Intelligence, D. Holmes and L. Jain, Eds. Springer Berlin Heidelberg, 2008, vol. 156, pp. 33–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85066-3_3

[59] Z. Karnin, E. Liberty, S. Lovett, R. Schwartz, O. Weinstein, S. Mannor, N. Srebro, and R. C. Williamson, “Unsupervised SVMs: On the complexity of the furthest hyperplane problem,” *Journal of Machine Learning Research*, vol. 23, pp. 1–18, 2012.

[60] V. Kecman and J. Brooks, “Locally linear support vector machines and other local models,” in *Proc. of the 2010 International Joint Conference on Neural Networks (IJCNN’10), Barcelona, Spain.* IEEE, July 2010, pp. 1–6.

[61] K. Abd Jalil, M. Kamarudin, and M. Masrek, “Comparison of machine learning algorithms performance in detecting network intrusion,” in *Proc. of the 2010 International Conference on Networking and Information Technology (ICNIT’10)*, June 2010, pp. 221–226.

[62] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, “A hybrid network intrusion detection framework based on random forests and weighted k-means,” *Ain Shams Engineering Journal*, vol. 4, no. 4, pp. 753–762, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2090447913000105

[63] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, September 1990.

[64] T. Moon, “The expectation-maximization algorithm,” *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, November 1996.

[65] J. A. Hartigan and M. A. Wong, “A K-means clustering algorithm,” *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[66] M. Ester, H. peter Kriegel, J. S, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proc. of the 1996 Knowledge Discovery and Data Mining Conferences (KDD’96), Portland, Oregon, USA.* AAAI Press, 1996, pp. 226–231.

[67] G. Cormode and S. Muthukrishnan, “What’s new: finding significant differences in network data streams,” in *Proc. of the 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM’04), Hong Kong, China*, vol. 3, March 2004, pp. 1534–1545 vol.3.

[68] A. Merlo, M. Migliardi, and P. Fontanelli, “On energy-based profiling of malware in android,” in *Proc. of the 2014 International Conference on High Performance Computing Simulation (HPCS’14), Bolgona, Italy*, July 2014, pp. 535–542.

[69] M. Curti, A. Merlo, M. Migliardi, and S. Schiappacasse, “Towards energy-aware intrusion detection systems on mobile devices,” in *Proc. of the 2013 International Conference on High Performance Computing and Simulation (HPCS’13), Helsinki, Finland.*, July 2013, pp. 289–296.

[70] I. Pillai, G. Fumera, and F. Roli, “F-measure optimisation in multi-label classifiers,” in *Proc. of the 21st International Conference on Pattern Recognition (ICPR’12), Tsukuba, Japan*, November 2012, pp. 2424–2427.

[71] G. Abuaitah and B. Wang, “Online data-centric anomaly detection framework for sensor network deployments,” in *Proc. of the 2014 International Conference on Computing, Networking and Communications (ICNC’14), Honolulu, Hawaii, USA*, February 2014, pp. 599–604.

[72] R. E. Schapire, “A brief introduction to boosting,” in *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI’99), Stockholm, Sweden.* Morgan Kaufmann Publishers Inc., July-August 1999, pp. 1401–1406.

[73] F. Barani, “A hybrid approach for dynamic intrusion detection in ad hoc networks using genetic algorithm and artificial immune system,” in *Proc. of the 2014 Iranian Conference on Intelligent Systems (ICIS’14), Bam, Iran*, February 2014, pp. 1–6.

[74] O. Salem, Y. Liu, and A. Mehaoua, “Anomaly detection in medical WSNs using enclosing ellipse and chi-square distance,” in *Proc. of the 2014 IEEE International Conference on Communications (ICC’14), Sidney, Australia*, June 2014, pp. 3658–3663.

[75] C. C. Holt, “Forecasting seasonals and trends by exponentially weighted moving averages,” *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207003001134

[76] H. Lauter, “Silverman, b. w.:density estimation for statistics and data analysis. chapman & hall, london - new york 1986, 175 pp.” *Biometrical Journal*, vol. 30, no. 7, pp. 876–877, 1988. [Online]. Available: http://dx.doi.org/10.1002/bimj.4710300745

[77] B. Kulis, “Metric learning: A survey,” *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.

[78] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, “Boltzmann machines: Constraint satisfaction networks that learn,” Computer Science Department, Carnegie Mellon University, Tech. Rep. CMU-CS-84-119, 1984.

[79] C. Guo, Y.-J. Zhou, Y. Ping, S.-S. Luo, Y.-P. Lai, and Z.-K. Zhang, “Efficient intrusion detection using representative instances,” *Computer & Security*, vol. 39, pp. 255–267, November 2013. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2013.08.003

[80] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, June 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390681.1442794

[81] P. Louvieris, N. Clewley, and X. Liu, “Effects-based feature identification for network intrusion detection,” *Neurocomputing*, vol. 121, pp. 265–273, December 2013.

[82] W. H. Kruskal and W. A. Wallis, “Use of ranks in one-criterion variance analysis,” *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1952.10483441

[83] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, “An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection,” *Applied Soft Computing*, vol. 12, no. 10, pp. 3285–3290, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494612002402

[84] M. Hayes and M. Capretz, “Contextual anomaly detection in big sensor data,” in *Proc. of the 2014 IEEE International Congress on Big Data (BigData’14), Anchorage, Alaska, USA*, June 2014, pp. 64–71.

[85] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Communication of the ACM*, vol. 51, no. 1, pp. 107–113, January 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[86] T. Sun and H. Tian, “Anomaly detection by diffusion wavelet-based analysis on traffic matrix,” in *Proc. of the 6th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP’14), Beijing, China*, July 2014, pp. 148–151.

[87] R. Long, W. Chen, and S. Yuan, “Wavelets generated by vector multiresolution analysis,” *Applied and Computational Harmonic Analysis*, vol. 4, no. 3, pp. 317–350, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1063520397902165

[88] R. R. Coifman and M. Maggioni, “Diffusion wavelets,” *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 53–94, 2006, special Issue: Diffusion Maps and Wavelets. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S106352030600056X

[89] T. Ahmed, B. Oreshkin, and M. Coates, “Machine learning approaches to network anomaly detection,” in *Proc. of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques, Cambridge, Massachusset (SYSML’07), Cambridge, Massachusetts, USA*. USENIX Association, April 2007.

[90] M. Xie, J. Hu, and S. Guo, “Segment-based anomaly detection with approximated sample covariance matrix in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2014.

[91] E. Bayraktar and L. Lai, “Byzantine Fault Tolerant Distributed Quickest Change Detection,” *arXiv:1306.2086*, December 2014. [Online]. Available: http://arxiv.org/abs/1306.2086

[92] E. S. Page, “Continuous Inspection Schemes,” *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954. [Online].

Available: http://dx.doi.org/10.2307/2333009

[93] E. Soltanmohammadi, M. Orooji, and M. Naraghi-Pour, “Decentralized hypothesis testing in wireless sensor networks in the presence of misbehaving nodes,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 205–215, January 2013.

[94] A. Rawat, P. Anand, H. Chen, and P. Varshney, “Collaborative spectrum sensing in the presence of byzantine attacks in cognitive radio networks,” *IEEE Transactions on Signal Processing*, vol. 59, no. 2, pp. 774–786, February 2011.

[95] K. Limthong, “Real-time computer network anomaly detection using machine learning techniques,” *Journal of Advances in Computer Networks*, vol. 1, pp. 1–5, 2013.

[96] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, “Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation,” in *Proc. of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX ’00), Hilton Head Island, South Carolina, USA*, vol. 2. IEEE, January 2000, pp. 12–26.

[97] J. Wang, D. Rossell, C. Cassandras, and I. Paschalidis, “Network anomaly detection: A survey and comparative analysis of stochastic and deterministic methods,” in *Proc. of the 52nd Annual Conference on Decision and Control (CDC’13), Florence, Italy*, December 2013, pp. 182–187.

[98] G. A. Carpenter and S. Grossberg, “ART 2: self-organization of stable category recognition codes for analog input patterns,” *Applied Optics*, vol. 26, pp. 4919–4930, December 1987.

[99] H. Egilmez and A. Ortega, “Spectral anomaly detection using graph-based filtering for wireless sensor networks,” in *Proc. of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP’14), Florence, Italy*, May 2014, pp. 1085–1089.

[100] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.

[101] K. Cohen and Q. Zhao, “Active hypothesis testing for quickest anomaly detection.” in *CoRR*, 2014.

[102] Q. Huang and P. P. C. Lee, “LD-Sketch: A distributed sketching design for accurate and scalable anomaly detection in network data stream,” in *Proc of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM’14), Toronto, Canada*. IEEE, May 2014.

[103] T. Bu, J. Cao, A. Chen, and P. P. Lee, “Sequential hashing: A flexible approach for unveiling significant patterns in high speed networks,” *Computer Networks*, vol. 54, no. 18, pp. 3309–3326, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610002045

[104] Y. Liu, W. Chen, and Y. Guan, “A fast sketch for aggregate queries over high-speed network traffic,” in *Proc. of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM’12), Orlando, Florida, USA*, March 2012, pp. 2741–2745.

[105] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, “Drebin: Efficient and explainable detection of android malware in your pocket,” in *Proc. of the 20th Annual Network & Distributed System Security Symposium (NDSS’14), San Diego, California, USA*, February 2014.

[106] W. Enck, M. Ongtang, and P. McDaniel, “On lightweight mobile phone application certification,” in *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS’09), Chicago, Illinois, USA*. ACM, 2009, pp. 235–245. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653691

[107] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Android permissions: A perspective combining risks and benefits,” in *Proc. of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT’12), Newark, New Jersey, USA*. ACM, 2012, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/2295136.2295141

[108] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy, “Using probabilistic generative models for ranking risks of android apps,” in *Proc. of the 2012 ACM Conference on Computer and Communications Security (CCS’12), Raleigh, North Carolina, USA*. ACM, 2012, pp. 241–252. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382224

[109] S. Liang and X. Du, “Permission-combination-based scheme for android mobile malware detection,” in *Proc. of the 2014 IEEE International Conference on Communications (ICC’14), Sidney, Australia*, June 2014, pp. 2301–2306.

[110] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proc. of the IEEE Symposium on Security and Privacy (SP'12), San Francisco, California, USA*, May 2012, pp. 95–109.

[111] W. Glodek and R. Harang, "Rapid permissions-based detection and analysis of mobile malware using random decision forests," in *Military Communications Conference, MILCOM 2013 - 2013 IEEE*, November 2013, pp. 980–985.

[112] S. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A new android malware detection approach using bayesian classification," in *Proc. of the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA'13), Barcelona, Spain*, March 2013, pp. 121–128.

[113] J. Sahs and L. Khan, "A machine learning approach to android malware detection," in *PRoc. of the 2012 European Intelligence and Security Informatics Conference (EISIC'12), Odense, Denmark*, August 2012, pp. 141–147.

[114] D. Damopoulos, G. Kambourakis, and G. Portokalidis, "The best of both worlds: A framework for the synergistic operation of host and cloud anomaly-based ids for smartphones," in *Proc. of the Seventh European Workshop on System Security (EuroSec'14), Amsterdam, The Netherlands*. ACM, April 2014, pp. 6:1–6:6. [Online]. Available: http://doi.acm.org/10.1145/2592791.2592797

[115] D. Damopoulos, G. Kambourakis, S. Gritzalis, and S. Park, "Exposing mobile malware from the inside (or what is your mobile app really doing?)," *Peer-to-Peer Networking and Applications*, vol. 7, no. 4, pp. 687–697, 2014. [Online]. Available: http://dx.doi.org/10.1007/s12083-012-0179-x

[116] D. Damopoulos, G. Kambourakis, and S. Gritzalis, "From keyloggers to touchloggers: Take the rough with the smooth," *Computers & Security*, vol. 32, pp. 102–114, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404812001654

[117] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Computers & Security*, vol. 43, pp. 1–18, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814000285

[118] Y. an Huang, W. Fan, W. Lee, and P. Yu, "Cross-feature analysis for detecting ad-hoc routing anomalies," in *Proc. of the 23rd International Conference on Distributed Computing Systems (ICDCS'03), Providence, Rhode Island, USA*, May 2003, pp. 478–487.

[119] R. Kohavi, "The power of decision tables," in *Proc. of the 8th European Conference on Machine Learning (ECML'95), Heraklion Crete, Grece*. Springer-Verlag, April 1995, pp. 174–189.

[120] B. Amos, H. Turner, and J. White, "Applying machine learning classifiers to dynamic android malware detection at scale," in *Proc. of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC'13), Cagliari, Italy*, July 2013, pp. 1666–1671.

[121] C. Wang, Z. Wu, A. Wang, X. Li, F. Yang, and X. Zhou, "Smartmal: A service-oriented behavioral malware detection framework for smartphones," in *Proc. of the 10th IEEE International Conference on High Performance Computing and Communications and Embedded and Ubiquitous Computing (HPCC_EUC'13), Zhangjiajie, China*, November 2013, pp. 329–336.

[122] R. Saaty, "The analytic hierarchy process—what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3–5, pp. 161–176, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0270025587904738

[123] M. Alam and S. Vuong, "Random forest classification for detecting android malware," in *Proc. of the 2013 Green Computing and Communications conference (GreenCom'13), Beijing, China*, August 2013, pp. 663–669.

[124] Y. Wei, H. Zhang, L. Ge, and R. Hardy, "On behavior-based detection of malware on android platform," in *Proc. of the 2013 IEEE Global Communications Conference (GLOBECOM'13), Atlanta , Georgia, USA*. IEEE, December 2013, pp. 814–819.

[125] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for android malware detection," in *Proc. of the 7th International Conference on Computational Intelligence and Security (CIS'11), Hainan, China*. IEEE, December 2011, pp. 1011–1015.

[126] K. J. Abela and al., "The analytic hierarchy process—what it is and how it is used," *International Journal of Cyber-Security & Digital Forensics*, vol. 2, no. 2, 2013. [Online]. Available: http://connection.ebscohost.com/c/articles/88222125/

automated-malware-detection-system-android-using-behavior-based-analysis-amda

[127] G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, “Madam: A multi-level anomaly detector for android malware,” in *Computer Network Security*, ser. Lecture Notes in Computer Science, I. Kotenko and V. Skormin, Eds. Springer Berlin Heidelberg, 2012, vol. 7531, pp. 240–253. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33704-8_21

[128] Y. R. Choe, M. Bierma, J. L. Erickson, D. J. Fritz, and E. Gustafson, *Andlantis: Large-scale Android Dynamic Analysis.* IEEE, February 2014, pp. 1–6.

[129] M. Zheng, M. Sun, and J. Lui, “Droidtrace: A ptrace based android dynamic analysis system with forward execution capability,” in *Proc. of the 2014 International Wireless Communications and Mobile Computing Conference (IWCMC’14), Nicosia, Cyprus*, August 2014, pp. 128–133.

[130] L. K. Yan and H. Yin, “Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis,” in *Proc. of the 21st USENIX Conference on Security Symposium (Security’12), Bellevue, Washington, USA*. USENIX Association, August 2012, pp. 29–29.

[131] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, “Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets.” in *Proc. of the 19th Annual Network & Distributed System Security Symposium (NDSS’12), San Diego, USA*. The Internet Society, 2012. [Online]. Available: http://dblp.uni-trier.de/db/conf/ndss/ndss2012.html#ZhouWZJ12

Author Biography

Sherenaz Al-Haj Baddar earned her PhD degree in Computer Science from Kent State University in Ohio (USA) in 2009. She obtained her M.Sc. and B.Sc. in Computer Science from the University of Jordan, in Amman, Jordan, back in 2003 and 2001 respectively. While pursuing her PhD, she worked with Prof. Kenneth E. Batcher on novel strategies for designing faster sorting networks. Currently, Dr. Al-Haj Baddar is an assistant professor of Computer Science at the University of Jordan. Her current research focuses on parallel and distributed computing and wireless sensor networks.

Alessio Merlo got a Laurea degree in Computer Science in 2005 at University of Genova. He received his PhD in Computer Science from University of Genova (Italy) in 2010 where he worked on performance and access control issues related to Grid Computing. He is currently serving as an Assistant Professor at the Computer Security Lab (CSec Lab) at DIBRIS, University of Genoa. His currently research interests are focused on security issues related to Web, distributed systems (BYOD), and mobile (Android platform).

Mauro Migliardi got his PhD in Computer Engineering in 1995. He was a Research Associate and Assistant Professor at the University of Genoa and Research Associate at Emory University as Co-PI in the HARNESS heterogeneous metacomputing project. Currently he is Associate Professor at the University of Padua and Supply Professor at the University of Genoa. He is also a member of the Scientific Committee of the Center for Computing Platforms Engineering and he has won the 2013 Canada-Italy Innovation Reward. His main research interest is distributed systems engineering in general; recently he focused on mobile systems, human memory support services, energy awareness and green security. He has tutored more than 80 among Bachelor, Master and PhD students at the Universities of Genoa, Padua and Emory, and he has authored or co-authored more than 100 scientific papers published in national and international, peer reviewed conferences, books and journals.