Quantitative Comparison of Unsupervised Anomaly Detection Algorithms for Intrusion Detection

Filipe Falcão*, Tommaso Zoppi[$], Caio Barbosa*, Anderson Santos*,
Baldoino Fonseca*, Andrea Ceccarelli[$], Andrea Bondavalli[$]
*Instituto de Computação, Universidade Federal de Alagoas, Brazil
E-mail: {filipebatista, cbvs, ass2, baldoino}@ic.ufal.br
[$]Dipartimento di Sistemi e Informatica, Universitá di Firenze, Italy
E-mail: {tommaso.zoppi, andrea.ceccarelli, bondavalli}@unifi.it

ABSTRACT

Anomaly detection algorithms aim at identifying unexpected fluctuations in the expected behavior of target indicators, and, when applied to intrusion detection, suspect attacks whenever the above deviations are observed. Through years, several of such algorithms have been proposed, evaluated experimentally, and analyzed in qualitative and quantitative surveys. However, the experimental comparison of a comprehensive set of algorithms for anomaly-based intrusion detection against a comprehensive set of attacks datasets and attack types was not investigated yet. To fill such gap, in this paper we experimentally evaluate a pool of twelve unsupervised anomaly detection algorithms on five attacks datasets. Results allow elaborating on a wide range of arguments, from the behavior of the individual algorithm to the suitability of the datasets to anomaly detection. We identify the families of algorithms that are more effective for intrusion detection, and the families that are more robust to the choice of configuration parameters. Further, we confirm experimentally that attacks with unstable and non-repeatable behavior are more difficult to detect, and that datasets where anomalies are rare events usually result in better detection scores.

Categories and Subject Descriptors

[**Security and privacy**]: Intrusion detection systems ; [**Computer systems organization**]: Dependable and fault-tolerant systems and networks

Keywords

Anomaly Detection, Intrusion Detection, Unsupervised Algorithms, Comparison, Attacks Datasets, Attack Model

1. INTRODUCTION

It is fully acknowledged that systems and networks are subject to cyber-attacks. Attackers may try to alter or disrupt services, ultimately leading to an adverse economic or safety impact. Amongst protection countermeasures, *Intrusion Detection Systems* (IDSs, [44]) were proposed to enhance network and system security. IDSs collect and analyze data from networks and systems indicators to detect malicious or unauthorized activities, based on the hypothesis that an ongoing attack has distinguishable effects on such indicators. Most of enterprise IDSs adopt *signature-based* detection algorithms [44], which consist of looking for predefined patterns (or "signatures") in the monitored data in order to detect an ongoing attack. Data is usually seen as a flow of data points, which represent observations of the values of the indicators at a given time. Signature-based approaches usually score high detection capabilities and low false positive rates when experimenting known attacks [15], but they cannot effectively adapt their behavior when systems evolve or when their configuration is modified. As an additional consequence, they are not meant to detect *zero day attacks*, which are novel attacks that cannot be matched to any known signature [35]. Moreover, when a zero-day attack that exploit newly added or undiscovered system vulnerabilities is identified, its signature needs to be derived and added as a new rule to the IDS [8].

To mitigate the problem above, *Anomaly-based* IDSs rely on anomaly detection algorithms, which are intended to find patterns in data that do not conform to the expected behavior of a system (or a network) [12]: these patterns are called *anomalies*. Anomaly-based IDS are built on the assumption that ongoing attacks will generate observable anomalies when monitoring performance indicators of the system [13] or network [27]. Potentially, they can adapt themselves to suit the current context of the system and ultimately detect novel attacks [10]. However, their detection efficacy is linked to the ability of characterizing expected and, consequently, anomalous behaviors. A poor characterization of the expected behavior of the system negatively impacts on the identification of malicious anomalous activities. As a consequence, anomaly-based detectors usually generate a higher number of false alarms than signature-based methods [12].

Different anomaly detection algorithms usually exhibit different rates of missed (*False Negatives*) and wrong detections (*False Positives*) and, consequently, have different detection capabilities. Although most of such algorithms have a generic, context-independent formulation, they are often more effective to detect specific attacks on specific systems or applications. Moreover, the manifestation of the anomaly

DOI 10.1145/3297280.3297314,
ISBN 978-1-4503-5933-7/19/04 $15.00.

is usually different from attack to attack and from system to system. In [12], anomalies are classified as i) *point anomaly, or outlier*: a single data instance that is out of scope or not compliant with the usual trend of an indicator, ii) *contextual anomaly*: a data instance that is unexpected in a specific context, and iii) *collective anomaly*: a collection of related data instances that is anomalous with respect to the entire data set. For example, when proposing angle-based algorithms [26], authors state that their technique suits the detection of point anomalies, rather than collective anomalies. Consequently, selecting the correct detection algorithm represents a crucial decision when defining an anomaly-based IDS. A wrong choice of the algorithm will decrease the attack detection capabilities of the IDS, consequently reducing the ability to secure the target system and network.

In this paper we first present a methodology for the experimental quantitative comparison of anomaly detection algorithms applied on multiple attacks datasets. We identify a total of 12 unsupervised anomaly detection algorithms that have been previously used for intrusion detection, two for each of the six (*clustering*, *statistical*, *classification*, *neighbour-based*, *density-based* and *angle-based*) families that are usually considered when grouping unsupervised anomaly detection algorithms [12], [21]. We adopt unsupervised anomaly detection algorithms since they are known to be the most suitable way to deal with evolving systems and to identify zero-day attacks [28].

Then, we identify 5 attacks datasets, namely KDD-CUP 99 [36], NSL-KDD [43], ADFA-LD [14], ISCX2012 [39], and UNSW-NB15 [32]. Since the datasets contain attacks labeled according to different nomenclatures, we categorize the attacks of the datasets in a *unified attack model*. Exercising the selected algorithms on the five datasets allows comparing the behaviour of the algorithms (both individually and as families) with respect to the datasets and the attacks of our attack model. Finally, we observe how an adequate distribution of expected and anomalous data points in the datasets helps improving detection scores, because it allows algorithms to properly define the expected behavior.

This paper is structured as follows: Section II presents a literature review of works on the comparison of anomaly detection algorithms for intrusion detection. Section III presents our methodology and inputs including the selection of the algorithms, the datasets, the attack model and the metrics that will be used for the experimental evaluation. Section IV presents the implementation of the algorithms and the setup of our experimental campaign. Section V discusses results. Lastly, Section VI concludes the paper.

2. RELATED WORKS AND MOTIVATION

Generally, the vast majority of research works on anomaly detection, and on anomaly-based IDSs, propose a novel technique for intrusion detection and then compare it with a small set of different algorithms executing on a single dataset. Valuable examples are cited throughout this paper, especially [26], [20], [23]. This structure is very effective in presenting a novel algorithm and showing how it performs compared to a few existing ones. However, the experimental comparison of the target algorithm with algorithms from the state of the art is often limited to proof-of-concept samples or few target datasets. A question that is often left open is if an algorithm that is proven effective for a given case study or dataset has a similar behaviour when applied to a different - although similar - context, or when it is evaluated using a different metric.

We believe that an extensive evaluation of algorithms by considering different categories of attacks, target systems (datasets), and scoring metrics would be beneficial. Being aware that the "silver bullet" algorithm, or rather an algorithm which always performs better than others, does not exist (yet?), we think that a deep comparison among *anomaly detection algorithms for intrusion detection* is needed to understand which (family of) algorithm is recommended when dealing with a specific class of attacks and systems.

2.1 On Comparing Anomaly Detectors

The authors of [25] used 7 algorithms on a single proprietary dataset containing HTTP traffic, providing an open-source IDS testing framework. Similarly, in [19] authors evaluate 4 algorithms on a single dataset, focusing more on feature selection. Instead, in [28], authors presented a comparative study for intrusion detectors where *k-Nearest Neighbors* (kNN), *Mahalanobis*-based, *Local Outlier Factor* (LOF) and one-class *Support Vector Machines* (SVM) were evaluated using only the DARPA 98 dataset [29] and real network data (for a total of 2 datasets). Similarly, in [17] authors compared three unsupervised anomaly detection algorithms for intrusion detection: *Cluster-based Estimation*, kNN and *one-class SVM* using network records stored in the *KDD Cup 99* dataset and system call traces from the *1999 Lincoln Labs DARPA evaluation*. Four algorithms are evaluated in [16], which presents a review of novelty detection methods that are classified into semi-supervised and unsupervised categories. The algorithms are exercised on 10 different datasets regarding medical and general-purpose data. Some of these datasets were used also in [21], where authors presented a comparison of anomaly detection algorithms for multivariate data points. In this case, 19 anomaly detection methods were evaluated in 10 different datasets from different domains, ranging from brain cancer to satellite activity; however, the only attacks dataset is the *KDD Cup 99*. Lastly, in [24] the authors compared six supervised and unsupervised algorithms for system log analysis using two datasets: the HDFS logs from Amazon EC2 platform and the BGL BlueGene/L supercomputer system at *Lawrence Livermore National Labs* (LLNL). In [25] seven anomaly detectors for IDSs are tested on HTTP traffic using a proprietary dataset.

2.2 Motivations

Summarizing, there are no extensive comparisons of (unsupervised) anomaly detection algorithms for intrusion detection. The works [21] and [16] considered multiple aspects that are also in this paper, but they do not focus on security and intrusion detection: in fact, they used datasets from multiple domains. Moreover, in [25] the authors considered a single proprietary dataset, while the work in [28] uses two datasets and four algorithms, without taking into account all the main families of algorithms defined in [12] and refined in [21]. Similarly, in [17] the authors used 3 algorithms on 2 datasets, while, [24] uses 3 unsupervised algorithms on 2 datasets. As a final remark, none of the reviewed papers organizes the results according to a unified attack model, that categorizes attacks from the different datasets.

To fill this gap, in this work we i) define a pool of 12 algorithms, selected from the 6 families of unsupervised al-

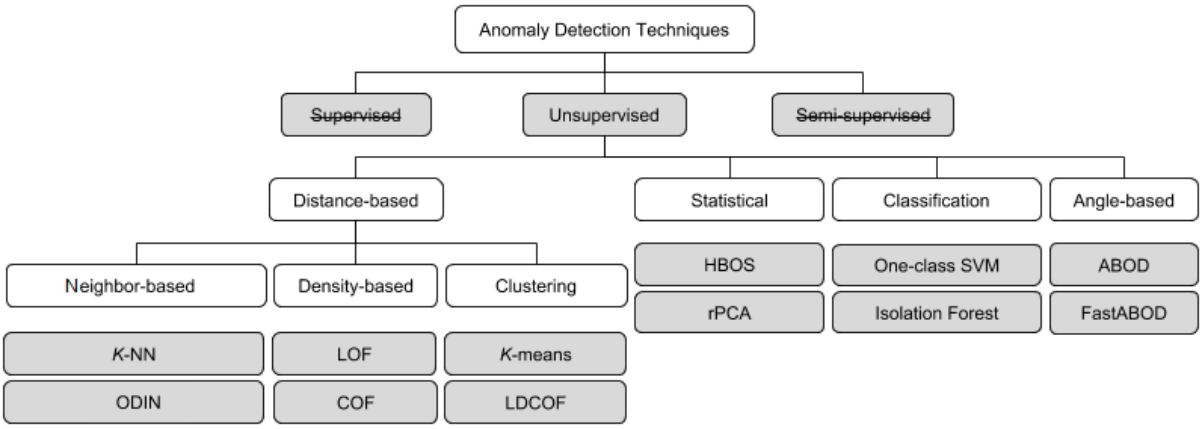


Figure 1: Classification of the 12 selected unsupervised anomaly detection algorithms from 6 families.

gorithms identified in [12], [21], ii) select a total of 5 publicly available attacks datasets, iii) define an unified attack model which categorizes attacks from the datasets above, iv) adopt the most used scoring metrics [34], and v) present our results, which are publicly available at [5] and constitute a baseline for comparing anomaly-based IDSs.

3. INPUTS FOR EXPERIMENTS

Our process is built on the methodology that is described in Section 4. Such methodology needs the following inputs: the anomaly detection algorithms (and their families), the selected datasets, a comprehensive attack model, and the metric(s) used for presenting results. These elements are expanded below.

3.1 Selection of the Algorithms

Several unsupervised anomaly detection algorithms were identified during our survey activity. A total of 12 unsupervised algorithms, reported in Figure 1, were selected that match the following criteria:

- the algorithms should not adopt (semi)supervised learning schemes, which are not adequate for adaptive systems and for identifying zero-day attacks. So we limit our analysis to unsupervised algorithms.
- for completeness, the set of algorithms should cover main families of unsupervised algorithms: *clustering, angle-based, statistical, neighbour-based, density-based, classification-based.*
- the algorithms should already have been successfully applied for intrusion detection.

The selected algorithms are mapped in Figure 1, according to their families from [12], [21]. It is worth noticing that algorithms families may have some unavoidable semantic overlap; as example, neighbour-based strategies are often used as a way to improve the detection capabilities of algorithms such as the angle-based *FastABOD* [26] algorithm.

Clustering

K-means Clustering (K-Means) [38] is a popular clustering algorithm that groups data points into k clusters by their feature values. First, the k cluster *centroids* are randomly initialized. Then, each data record is assigned to the cluster with the nearest centroid, and the centroids of the modified clusters are re-calculated. This process stops when the centroids are not changing anymore. Data points are put in the same cluster when they have similar feature values according to a given distance function. Finally, scores of each data point inside a cluster are calculated as the distance to its centroid. Data points which are far from the centroid of their clusters are labeled as anomalies.

Local Density Cluster-based Outlier Factor (LDCOF) [6] estimates the density of clusters generated by using the K-means clustering algorithm, which are separated into small and large groups following the procedure of [24]. For each cluster, the average distance of all its data points to the centroid is calculated, normalized by the average distance of the data points of this cluster to its centroid, and used as anomaly score. Therefore, expected data points result in smaller scores i.e., close to 1.0, because their densities are as big as the densities of their cluster neighbors. Instead, anomalies will result in larger scores, since their densities are smaller than the densities of their neighbors.

Neighbor-Based

Kth-Nearest Neighbor (kNN) [35] is a *neighbour-based* method which was primarily designed to identify outliers. For each data point, the whole set of data points is examined to extract the k items that have the most similar feature values: these are the k nearest neighbors (NN). Then, the data point is classified as anomalous if the majority of NN was previously classified as anomalous. Note that while the nearest neighbours are gathered in an unsupervised manner, the final classification needs some labels in the training set.

Outlier Detection using Indegree Number (ODIN) [23] stems from kNN, which examines the whole dataset to determine their feature distances to the given point. This allows isolating NN, creating the kNN graph. Differently from kNN, ODIN defines as anomalies the data points that have a low number of in-adjacent edges in the kNN graph.

Density-Based

Local Outlier Factor (LOF) [9] is a *density-based* method designed to find local anomalies. For each data point, the NN are computed. Then, using the computed neighbor-

hood, the local density is computed as the *Local Reachability Density (LRD)*. Finally, the LOF score is computed by comparing the LRD of a data point with the LRD of the previously computed NN. As LDCOF, data points classified as expected by LOF will have smaller scores, close to 1.0, and data points classified as anomalies will have larger scores.

Following the approach of local density, **Connectivity-based Outlier Factor (COF)** [41] differs from LOF in the computation of the density of the data points, since it also considers links between data points. To such extent, this method adopts a shortest-path approach that calculates a *chaining distance*, using a *minimum spanning tree* [10].

Statistical

The **Histogram-based Outlier Score (HBOS)** is a *statistical* approach [20] that generates an histogram for each independent feature of the given dataset. The values of the features of all the available data points are first used to build histograms; at a later stage, for each data point, the anomaly score is calculated as the multiplication of the *inverse heights* of the columns in which each of its features falls.

Robust Principal Component Analysis (rPCA) [27] is based on the Principal Component Analysis (PCA), that is used for dimensionality reduction. PCA is used to detect subspaces in a dataset and has been applied to anomaly detection to identify deviations from the 'expected' subspaces, which may indicate anomalous data points. The principal components of PCA are the *eigenvectors* of the covariance matrix, which is computed twice to improve robustness.

Angle-Based

Angle-Based Outlier Detection (ABOD) [26] relates data to high-dimensional spaces, using the variance in the angles between a data point to the other points as anomaly score. Each data point in the dataset is used as the middle point p_2 of a polygonal chain (p_1, p_2, p_3), while p_1 and p_3 are any two different data points of the dataset, $p_1 \neq p_2 \neq p_3$. Then, all the angles $p_1\hat{p_2}p_3$ are measured, and their variance is used to calculate the *Angle-Based Outlier Factor* (ABOF). Ultimately, anomalies typically result in very small variance in the angles from couples of points.

The **Fast Angle-Based Outlier Detection (FastABOD)** [26], similarly to ABOD, detects anomalous data points depending on the variance of angles between pairs of distance vectors to other points. However, it works in subquadratic time by considering only angles between pairs of neighbours. For each data point, the algorithm first calculates the ABOF to its k-nearest neighbor as the normalized scalar product of the difference vectors of any pair of neighbors. Then, FastABOD ranks the data points according to their ABOF. The smaller the ABOF, the bigger the probability that the data point is anomalous.

Classification-Based

The **One-class Support Vector Machine (one-class SVM)** [7] algorithm aims at learning a decision boundary to group the data points [37]. Therefore it *can be used for unsupervised anomaly detection*, despite at first supervised support vector machines (SVMs) were used only for semi-supervised anomaly detection [12]. The one-class SVM is trained with the dataset and then each data point is classified considering the normalized distance of the data point from the determined decision boundary [7].

Table 1: Selected Data sets

Dataset	Size	Attacks	% Attacks	Features
KDD Cup 99 (KC)	311,028	223,298	71.79	41
NSL-KDD (NK)	22,542	12,832	48.05	42
ISCX2012 (IX)	571,698	66,813	11.68	17
ADFA-LD (AL)	2,122,085	238,160	11.22	1
UNSW-NB15 (UN)	175,341	119,341	68.06	46

Isolation Forest (IF) [30] structures data points as nodes of an isolation tree, assuming that anomalies are rare events with feature values that differ a lot from expected data points. Therefore, anomalies are more susceptible to isolation than the expected data points, since they are isolated closer to the root of the tree instead of the leaves. It follows that a data point can be isolated and then classified according to its distance from the root of the tree.

3.2 Selection of the Datasets

The datasets are selected to match the following criteria:

- They shall contain enough data points to ensure statistical evidence when evaluating the algorithms, e.g., DARPA 1999 dataset [29] was discarded since it contains only 201 data points related to attacks.
- They shall be labeled correctly, i.e., all and only attacks that occurred should appear. Consequently, we disregard datasets as MAWI [18], CAIDA [39] or DEFCON [39], that are constituted of sniffed data that is labeled applying detection algorithms: consequently, the labeling may include false positives and negatives.
- Data points should be complete for all the features in the datasets, to do not apply feature recovery strategies that may inficiate results.

Table 1 summarizes the 5 datasets we selected according to these criteria. These datasets are shortly described below by ascending publication date. We match each dataset to an acronym that will be used in the rest of the paper. We also remark that filtering or refinement of datasets is usually algorithm-specific [12], (e.g., filtering massive amounts of subsequent outliers [38], or incomplete data points [17], [41]); therefore we do not modify the datasets in any way. As a side note, during our selection process we also found two datasets, Kyoto2006+ [40] and NGIDS-DS [22], which we discarded since the amount of data was too huge to be processed in a meaningful percentage.

KC *KDD Cup 99 (1999)* [36]. This is the most popular dataset in the anomaly-based intrusion detection area, being used in recent experiments and surveys [10], [21] and works prior the release of the updated NSL-KDD [23]. Fore this reason, we could not ignore it despite being almost 20-years-old. The dataset contains the following attacks: *DoS* (Denial of Service), *R2L* (unauthorized access from a remote machine), *U2R* (unauthorized access to *superuser* or *root* functions) and *Probing* (gather information about a network).

NK *NSL-KDD (2009)* [43]. This dataset was created to solve problems in the *KDD Cup 99* dataset as i) the presence of redundant records in train sets, and ii) duplicates in test sets. The attacks are the same as KC.

Table 2: The Unified Attack Model developed to categorize attacks from the five datasets.

Category	Description	Mapping of (Dataset) Attack
Communication - Passive	Attacks which targets the communication channel to gather information without active damage	(KD - NK) Probing, (IX) Infiltration, (UN) Reconnaissance, (UN) Analysis
Communication - Active	Attacks conducted through the communication channel to actively damage the system	(IX) Bruteforce, (KD - NK - IX - UN) DoS, (IX) DDoS, (UN) Fuzzers, (UN) Backdoor
Host	Attacks which targets a given host by installing malicious code into it	(KD - NK) U2R, (KD - NK) R2L, (UN) Worms, (UN) Shellcode, (UN) Malware
Application	Attacks which targets a given application aiming at executing malicious code by penetrating interfaces	(UN) Exploits, (UN) Generic, (AL) AddUser (AL) Java, (AL) Meterpreter, (AL) Web, (AL) Hydra

IX *ISCX (2012)* [39]. It is generated in a controlled environment based on a realistic network and traffic, to depict the real effects of attacks over the network and the corresponding responses of workstations. Four different attack scenarios are simulated: *infiltration*, *HTTP denial of service*, a *distributed denial of service* by using an IRC botnet, and SSH *bruteforce* login attempts.

AL *ADFA-LD (2013)* [14]. Released by the Australian Defence Force Academy, this dataset contains expected and anomalous *Linux system call* traces generated by emulation. The occurrence of *AddUser*, *Java*, *Meterpreter*, *Hydra SSH-FTP*, and *Web* attacks is labelled, although not detailed.

UN *UNSW-NB15 (2015)* [32]. As ADFA-LD, this dataset was released by the *Australian Defence Force Academy* in the *University of New South Wale*. Authors simulate: i) *Exploits*, the attacker exploits a generic vulnerability, ii) *DoS*, a (Distributed) Denial of Service, iii) *Worms*, a script that replicates itself to spread to other networked computers, iv) *Generic*, a technique that works against all block-ciphers, with a given block and key size, v) *Reconnaissance*, attack that aim at gathering information, vi) *Shellcode*, a code used as the payload in exploits, and vii) *Backdoors*, that stealthily bypass security mechanisms to access data.

3.3 Unified Attack Model

Each of the datasets above uses inconsistent naming and grouping of categories of attacks: consequently, to perform cross-datasets comparisons, we have to define an attack model that includes all the attacks in the different datasets. To such extent, we classify the attacks of the different datasets according to a *unified attack model* that we derived after examining several standards [42], [11], open source libraries [3], and research articles [33], [31] providing taxonomies of cyber-attacks that are largely adopted by topic-related studies. Noticeably, in [33] the authors partition the attacks defined in the *NIST 800-53* [42] standard into i) *Communication*: attacks directed to network interfaces, ii) *Host*: malware or malicious code injected in a target host exploiting vulnerabilities of the operating system, iii) *Application*: attacks that exploits vulnerabilities of (web)services, and iv) *Generic*, everything that is not related to the first three categories. The work [33] provides a simplified abstraction of the existing *NIST* attack list, which we can use as a starting point to build our unified attack model.

After an in-depth revision of the attack models adopted by each dataset, we can observe that none of the specific attacks falls in the Generic category of [33], which is consequently discarded. However, the datasets contain data related to attacks that mainly involve the communication channel with different means and purposes. To differentiate among these attacks, we split the *Communication* category defined in [33] into two separate categories (see row 1 and 2 of Table 2). Namely, the subcategories are i) *communication passive*, that represents attacks directed to gather or steal data through the passive observation of the communication channel, and ii) *communication active*, which represents attacks which use the communication channel as a way to send malicious data / requests to the target system. This ultimately results in an attack model composed of 4 categories, and that is shown in Table 2. The last column of Table 2 maps all the different attacks referenced in the datasets to our attack categories. As example, *Exploits* attacks of UNSW-15 fall into the *Application* category, as it can be observed at the bottom right cell of the table. Attacks with different labels, or reported in different datasets, that resemble the same attack are merged into a unique attack, e.g., DoS, which can be found in both NK and IX datasets.

3.4 Selection of the Metrics

We list here the metrics described in [34] that were used more frequently in the surveyed studies. These metrics are mainly based on boolean *anomaly/expected* labels assigned to a given data point. However, when providing an output, algorithms may not provide a *boolean* answer: instead, they usually provide a numeric anomaly score, which indicates how anomalous a data point is in relation to the others. In our study, we define such thresholds relying on the *Interquartile Range*, that is the difference of the two *quantiles* $Q3$ and $Q1$ that was extensively studied in [45], and adopted as a recommended practise when dealing with numerical data, as in [27]. In addition, we do not account for time-related metrics as the execution time and the detection time, because they are usually dependent on the hardware resources available for a given system and on the specific implementation of the algorithm.

- **Precision (P).** Considering TP (true positives) as the amount of detected anomalies that correspond to manifestations of attacks, and FP (false positives) as the amount of detected anomalies that do not, *Precision* is defined as the fraction of TPs among the union of FPs and TPs.

- **Recall (R).** Recall is usually presented together with P. It is defined as the ratio of TP over the union of TP and the undetected anomalies (false negatives, FN).

- **F-Score (F_β) and F-Measure.** The $F - Score(\beta)$ metric combines both P and R by using a parameter β: when $\beta > 1$, R is weighted more than P. Instead, F-Measure, or rather F_1, is defined as the balanced mean of P and R [34] and is adopted when data analysts evaluate FPs and FNs as equally undesired.

- **Accuracy (ACC).** Accuracy is defined as the ratio of correct detections (both true positives and true negatives) among the total amount of examined data points. This allows aggregating the positive and negative scores into a unique metric.

- **Area Under ROC curve (AUC).** *ROC curve* is a graphical plot that represents the performance of binary classifiers when their discrimination thresholds vary: the curve is depicted by plotting R against the false positive rate. An high value of the area underlying the ROC curve usually indicates that the identified algorithm suits the target dataset.

4. METHODOLOGY, IMPLEMENTATION AND EXECUTION

Before executing the experiments, we partition each dataset to create sub-datasets isolating single types of attacks. Then, we apply the 12 algorithms on each sub-dataset separately. The execution of each algorithm on each sub-dataset is repeated by combining the parameters of the algorithm itself: such parameters are algorithm-specific, and require an initial setup to effectively run the algorithm. Boolean results of algorithms can be immediately used to calculate the metrics, while numeric scores have to be transformed by using adequate thresholds. Metric scores are then stored in a database and aggregated to extract average, median and standard deviation scores depending on different dimensions of analysis, namely: algorithm, algorithm family, attack, attack category, and dataset. To present aggregated results, we mostly use the format *median* $\pm$ *standard deviation*.

Then, we retrieve available public implementations of the selected algorithms. *KMeans*, *kNN*, *ODIN*, *LOF*, *COF*, *ABOD*, *FastABOD* and *OneClass SVM* are extracted from the ELKI framework [1]. The other implementations of algorithms come from RapidMiner [4]. The 5 datasets are processed by converting - where needed - nominal variables to numerical to increase the amount of usable features, without affecting the semantics of the datasets. Parameters tuning is employed to find an optimal setup of each algorithm. Tuning is performed by i) first, running different combinations of parameters; ii) then, comparing results for the different parameters. For example, we run KNN and KNN-dependent algorithms, i.e., ODIN, FastABOD, with $k \in \{1, 2, 3, 5, 10, 20, 50, 100\}$. The discussion in the following sections will also elaborate on the relevance of the choice of parameters.

The metric scores generated by the execution of the algorithms was initially stored in CSV files, and successively in a *MongoDB* [2] database for fast analytics, such as comparing and aggregating scores. Ultimately, we plot ROC curves, and calculate the *Area under the Curve* (AUC). The algorithms, the database and the *Mongo* analytics are executed on a cluster of three Intel Core i7, 32GB of RAM and 256GB of SSD storage servers. Overall, running all the algorithms on all the datasets required approximately one month of 24H execution. As pointed out by authors of angle-based algorithms [26], due to known computational complexity problems ABOD and FastABOD algorithms can be ran only on a portion of the datasets. We choose the portions considering the biggest subset of the dataset that do not escalate in heap memory errors, i.e., 4% for KC, 53% for NK, 24% for AL, 5% for IX, 6% for UN. All the metric scores are publicly available on our website [5].

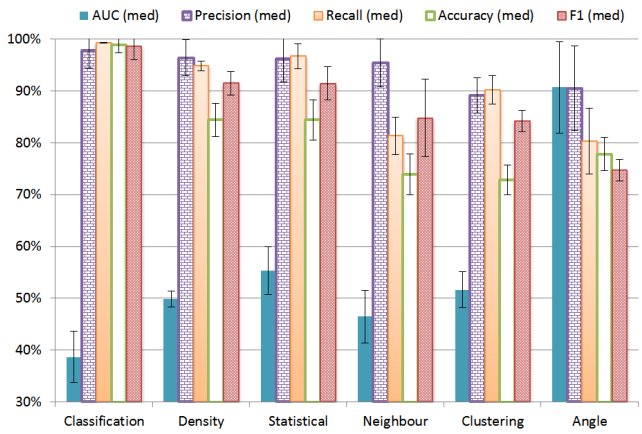


Figure 2: Results on all the datasets and all the attacks, grouped by algorithms families. Columns report median scores, while error bars depict the standard deviation

5. RESULTS AND DISCUSSION

We present our analysis by answering 4 *research questions*. Each research question is discussed and substantiated by referring to the results we obtained during our experimental campaign. A complete view of the collected data cannot be presented here for brevity: however, the *MongoDB* archive complemented with *Excel* files we used to derive graphs are available at [5].

5.1 RQ1: Is there an algorithm (or a family) that performs better than the others?

Table 3 shows the results we obtained by running all the 12 algorithms in our 5 datasets, ranked by F1 scores. We report the median and the standard deviation scores for each metric, and we aggregate the data by considering each algorithm on all the attacks and all the datasets separately. We observe that the first two algorithms belong to the *classification* family. In fact, both *Isolation Forest* and *OneClass SVM* show good scores for anomaly detection: Precision, Recall and Accuracy scores are above 96%. Opposed to classification algorithms, *angle-based* algorithms show poor results for the F1 scores on our datasets. This may be partially explained considering that training of such algorithms was performed by using just a portion of the datasets, negatively affecting their ability to characterize an expected behavior and - consequently - highlight anomalies.

Another interesting observation is represented in Figure 2, where we aggregate and plot the results related to each family. Results are again ordered by F1 score, in descending order. The *classification* family is the most effective, while *statistical* and *density-based* families show very similar scores. Moreover, it is worth noticing that *neighbour-based*

Table 3: Metric scores ($median \pm std$) for the 12 algorithms, ordered by $F1$ score

Algorithm	# Combinations	Family	AUC	Precision	Recall	Accuracy	F1
Isolation Forest [30]	8	Classification	37.2 ± 0.4	99.9 ± 0.3	99.3 ± 0.4	99.7 ± 0.3	99.6 ± 0.3
One-Class SVM [7]	1	Classification	53.4 ± 2.9	96.6 ± 3.2	99.3 ± 0.0	96.2 ± 3.2	98.0 ± 1.9
COF [41]	8	Density-Based	48.8 ± 1.7	93.6 ± 3.4	97.8 ± 0.1	91.7 ± 3.1	95.7 ± 2.0
ODIN [23]	8	Neighbour-Based	49.9 ± 1.7	96.6 ± 2.4	99.9 ± 0.4	89.8 ± 1.6	94.6 ± 1.1
HBOS [20]	1	Statistical	57.8 ± 5.5	92.6 ± 5.8	99.5 ± 4.3	89.2 ± 4.7	94.3 ± 4.8
rPCA [27]	1	Statistical	55.0 ± 4.0	97.5 ± 3.4	95.0 ± 1.0	83.1 ± 3.2	90.6 ± 2.0
LOF [9]	8	Density-Based	50.0 ± 1.3	96.6 ± 3.5	88.0 ± 1.1	81.3 ± 3.1	89.6 ± 2.1
LDCOF [4]	8	Clustering	49.9 ± 2.3	82.4 ± 1.8	94.4 ± 0.2	77.9 ± 1.5	87.4 ± 0.7
KNN [35]	8	Neighbour-Based	35.9 ± 6.7	91.9 ± 5.8	75.1 ± 3.4	71.4 ± 4.0	82.8 ± 4.3
K-Means [38]	8	Clustering	54.4 ± 8.9	95.7 ± 5.3	68.5 ± 2.8	65.6 ± 3.4	78.3 ± 3.5
ABOD [26]	8	Angle-Based	90.5 ± 7.8	69.2 ± 8.1	92.4 ± 8.3	90.0 ± 1.8	75.5 ± 10.2
FastABOD [26]	15	Angle-Based	86.4 ± 9.2	90.6 ± 7.8	77.4 ± 5.3	67.6 ± 3.2	74.7 ± 6.1

scores are a bit lower than the previously mentioned families, but with a higher standard deviation. In fact, it can be observed that there is a big difference of scores between the two *neighbour-based* algorithms: KNN is significantly worse than ODIN, which instead shows a remarkably high recall score. At first, this result surprised us, but it can be explained by the fact that ODIN is based on the KNN graph with some 'indegree score'. This semi-density score which is added on the top of the KNN query provides a decisive support to improve the detection scores, as also was anticipated in [23]. A similar situation can be observed also in the *clustering* family: the KMeans algorithm is used as a baseline for the LDCOF strategy, which shows far better scores on our datasets. Our last observation is on Accuracy scores. Here we can see how *angle-based* algorithms, which overall showed the worst F1 scores, have higher Accuracy values than the neighbour-based and clustering families. This can be motivated as follows: F1 score is based on Precision and Recall, which do not account for true negatives. As a result, higher Accuracy scores for angle-based algorithms compared to the corresponding F1 scores highlight that the percentage of true negatives is higher than the others.

5.2 RQ2: Is there an algorithm (or a family) that is less dependent on the choice of parameters?

This RQ is related to the choice of the optimal parameters of the algorithms, and it can be explained with the aid of Table 3 and Figure 2. In particular, the scores we used to build the table and to plot the graph refer to the median scores related to the best parameter setup for a given algorithm. Each algorithm has its own parameters, e.g., the size of neighbourhood k in KNN, ODIN and FastABOD. Such parameters have to be explored to find an optimal setup, in order to use the algorithms at the best of their detection capability. To evaluate the impact of such parameters, our methodology in Section 4 repeats the experiments using the same algorithm, but with different parameters values. The number of combinations of acceptable parameters values are reported in Table 3. These combinations are used to build the ROC Curve and, consequently, the AUC score. As expected, such score is low for classification algorithms (see Figure 2), which have several configurable parameters. This has strong consequences on our analysis: despite these algorithms show a very low number of FPs and FNs, i.e., accuracy and F1 are almost perfect,*classification algorithms strongly depend on their parameters values, and therefore cannot be always considered as an optimal solution.*

Surprisingly, it turns out that angle-based scores are not heavily influenced by the choice of such parameters, showing very high AUC scores, i.e., median over 85%, compared to the others which are instead mostly around 50%. This remarkable difference can be explained as a combination of two factors. First, and more important, such algorithms have few configurable parameters, and the way the ABOF is calculated makes them structurally robust to a wrong choice. Second, such big difference could also be linked to the selection of possible thresholds and parameters values: the AUC scores is high if changing such values does not significantly impact on the outcome of the analysis. Indeed, we remark that we selected the combinations of parameters and thresholds following the same methodology - analyze what authors wrote on parameters selection when presenting their algorithms - for all the algorithms.

5.3 RQ3: Is there an attack (or a category of attacks) that is more difficult to detect than the others?

We consider the attacks, or the attack categories, as dimensions of the analysis. This allows discussing the median scores of our 12 algorithms, for the anomalies generated by each specific attack and category.

5.3.1 Attack Analysis

Figure 3 shows P, R, ACC and F1 scores related to each of the attacks in the considered datasets. From left to right of Figure 3, it is possible to observe attacks with decreasing F1 scores. On the right of the figure, we depict the attacks that turned out to be tricky to detect, and that generated the highest amount of FPs and FNs. More in detail, it is possible to observe that anomalies generated by the *generic* and *exploits* attacks are difficult to detect. In fact, these attacks have heterogeneous characteristics: it is not trivial to define an expected behaviour, and consequently it is difficult to define what is not expected, i.e., anomalous. Another interesting observation regards *DoS* and *DDoS* attacks: Fig-

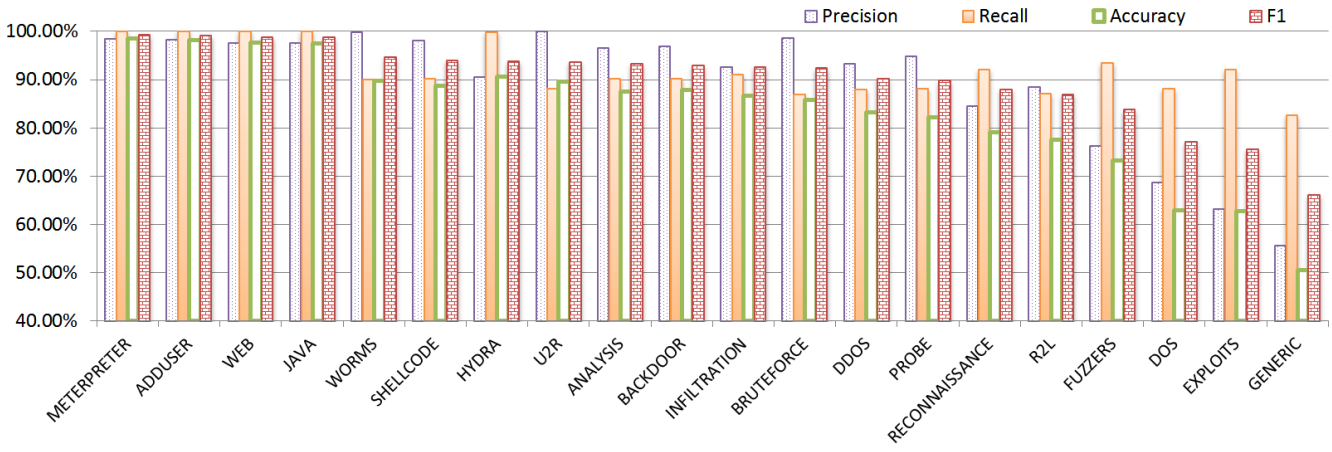


Figure 3: Median metric scores for all the datasets and algorithms, grouped by attacks.

ure 3 shows that median scores for DDoS are higher than DoS, implying that the anomalies generated by DDoS attacks are detected with higher probability. When multiple malicious hosts are trying to flood the target system with network packets, the packets arrival rate raises significantly, and therefore it is noticeably different from the expected rate. On the other side, anomalies due to *Meterpreter*, *AddUser*, *Web* and *Java* attacks are detected with excellent R scores (100%), implying that no false negatives were generated. We can conclude that most of the *Application* attacks are on average easily detected by the algorithms we considered in our study, while it is more difficult to detect *generic* and *exploit* attacks, probably because they introduce minimal - and not homogeneous - perturbations of the system and network behaviour.

5.3.2 *Categories of Attacks*

This result is confirmed by Figure 4(a) and 4(b), which plot R and ACC median scores for each combination of the 4 attack categories and the 6 algorithms families. The two scatter plots depict the same 24 combinations of attack categories and algorithms families. To elaborate on attacks, in Figure 4(a) 4 different symbols are assigned to each attack category, while Figure 4(b) contains 6 different series of symbols that help examining algorithms families. The application attacks (the red circles in Figure 4(a)) are on average very close to the right border, meaning that high R scores were reported for all the algorithms families. Figure 4(a) highlights the good performances of classification algorithms we already discussed in RQ1. In addition, we point out that algorithms belonging to the statistical, classification and density-based families have high R scores for each algorithm, i.e., crosses, segments and diamonds are positioned on the right side of Figure 4(b), in the 10th best percentile. Moreover, we can observe how ACC is always over 70%, except for *host* attacks (see Figure 4(a)) with angle-based algorithms (bottom left of Figure 4(b)).

5.4 RQ4: Is there a dataset that offers better detection scores than others?

The question is mostly related to the intrinsic difficulties of correctly defining an expected behaviour relying on the available data points of a given dataset. Since anomalies are supposed to be rare events, our conjecture is that in datasets with a low ratio of attacks it should be easier to define the expected behaviour and, consequently, to detect anomalies.

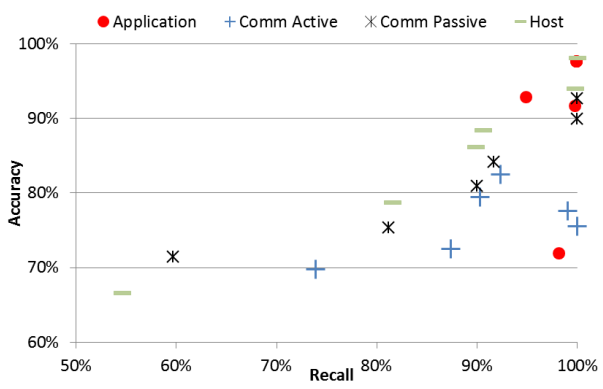
Table 4: Metric scores ($median \pm std$) for each dataset

Dataset	AUC	Recall	Precision	Accuracy	F1
KC [36]	44.1 ± 8.2	95.3 ± 11.4	86.2 ± 4.9	80.0 ± 7.2	88.8 ± 8.4
NK [43]	48.7 ± 8.6	87.6 ± 3.2	88.2 ± 2.5	75.3 ± 2.6	84.6 ± 2.2
IX [39]	45.8 ± 3.4	97.4 ± 0.5	88.6 ± 1.5	86.0 ± 1.4	92.4 ± 0.7
AD [14]	49.9 ± 1.4	97.6 ± 1.0	99.4 ± 0.7	97.5 ± 1.4	98.8 ± 0.5
UN [32]	49.8 ± 3.2	84.8 ± 3.5	90.2 ± 3.1	74.0 ± 3.3	84.5 ± 2.6

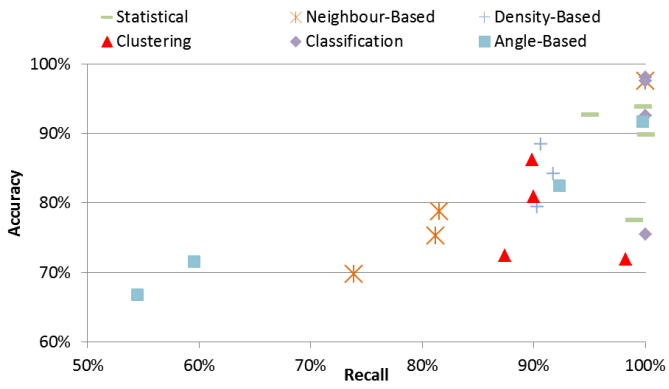
Table 4 aggregates the results of all the algorithms for all the attacks contained in each dataset, partially confirming this conjecture. From the top of the table, we can see that the higher F1 scores are related to the AL and the IX datasets: according to Table 1, these contain respectively 11.22% and 11.68% of attacks. However, this reasoning does not apply to the KC and NK datasets: KC has an higher ratio of attacks than NK, but it has better scores than NK. This may also be related to the specific attacks in the datasets: some of them may be easier to detect, because they generate point anomalies that significantly differ from the expectations. In addition, it is worth noticing that the AUC resulted in very similar and poor scores (between 40% and 50%) across all the datasets, meaning that thresholds and parameters of the algorithms should be tuned carefully before conducting anomaly detection on the datasets we selected for this study.

6. CONCLUSIONS AND FUTURE WORKS

This paper aims at providing a baseline reference for quantitative comparison of anomaly-based intrusion detectors for critical distributed systems, which may be used by researchers and practitioners when designing and assessing anomaly-based IDSs. We compared 12 anomaly detection algorithms belonging to 6 families and 5 attacks datasets. Assuming that ongoing attacks manifest unstable performances of system resources and processes, we exercised each algorithm on each dataset, collecting a database of metric results which is publicly available at [5]. We aggregated, presented and discussed our results by means of research questions. Main find-



(a) Algorithms Families and *Attack Categories*. The graph shows 4 series, one for each attack category.



(b) *Algorithms Families* and Attack Categories. The graph shows 6 series, one for each algorithms family.

Figure 4: Median scores of Recall and Accuracy metrics for algorithms families and attacks categories. The graphs contain 24 items, each of them maps the scores of a given algorithms family to a specific attack category

ings are that algorithms belonging to the *classification* family showed overall better scores on the 5 selected datasets, while *angle-based* algorithms turned out to be more robust to the choice of the initial parameters. We also highlighted how the majority of attacks in the *application* category were detected easier than others, and how an adequate distribution of expected and anomalous data points in the datasets generally helps improving detection scores.

The results above suggest that *many works may be built on this baseline, contributing to anomaly-based intrusion detection*. In particular, our current and future works aim at understanding if *some algorithms may cover large sets of attacks*, while the uncovered attacks may be detected by using other algorithms in conjunction with the primary strategy. Then, we will investigate *if specific algorithms or algorithms families are particularly effective in detecting either point, contextual or collective anomalies*. The analysis may be complemented by deriving which kind of anomaly our (category of) attacks are more likely to generate. As a final result, a *link between attacks and algorithms* would be obviously defined, consolidating the selection of unsupervised algorithms for IDSs, also for complex systems [46].

Acknowledgments

This paper has been partially supported by the European FP7-IRSES DEVASSES and by the H2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement 823788 (ADVANCE project).

7. REFERENCES

[1] Elki data mining. `elki-project.github.io/`. Accessed: 2018-09-04.

[2] Mongodb for giant ideas. `mongodb.com/`. Accessed: 2018-09-04.

[3] Open web application security project (owasp). `https://www.owasp.org/index.php/Main\_Page`. Accessed: 2018-09-04.

[4] Rapidminer - anomaly detection. `github.com/Markus-Go/rapidminer-anomalydetection`. Accessed: 2018-09-04.

[5] Repository for additional files. `https://github.com/tommyippoz/Miscellaneous-Files/blob/master/SAC2018_SupportFiles.rar`. Accessed: 2018-12-12.

[6] M. Amer and M. Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer.

[7] M. Amer, M. Goldstein, and S. Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15. ACM, 2013.

[8] L. Bilge and T. Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844. ACM, 2012.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[10] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. On the evaluation of outlier detection: Measures, datasets, and an empirical study continued. In *Lernen, Wissen, Daten, Analysen 2016*. ceur workshop proceedings, 2016.

[11] T. Casey. Threat agent library helps identify information security risks. 2007.

[12] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[13] D. Cotroneo, R. Natella, and S. Rosiello. A fault correlation approach to detect performance anomalies in virtual network function chains. In *Software Reliability Engineering (ISSRE), 2017 IEEE 28th Int. Symposium on*, pages 90–100. IEEE, 2017.

[14] G. Creech and J. Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 4487–4492. IEEE, 2013.

[15] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz. An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks. *Expert systems with Applications*, 29(4):713–722, 2005.

[16] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire. An experimental evaluation of novelty detection methods. *Neurocomputing*, 135:313–327, 2014.

[17] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the Int. Conference on Machine Learning*. Citeseer, 2000.

[18] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda. Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th Int. Conference*, page 8. ACM, 2010.

[19] V. Garcia-Font, C. Garrigues, and H. Rifà-Pous. A comparative study of anomaly detection techniques for smart city wireless sensor networks. *Sensors*, 16(6):868, 2016.

[20] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. 2012.

[21] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[22] W. Haider, J. Hu, J. Slay, B. Turnbull, and Y. Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Appl.*, 87:185–192, 2017.

[23] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier detection using k-nearest neighbour graph. In *Pattern Recognition, ICPR 2004. Proceedings of the 17th Int. Conference on*, volume 3, pages 430–433. IEEE, 2004.

[24] S. He, J. Zhu, P. He, and M. R. Lyu. Experience report: system log analysis for anomaly detection. In *Software Reliability Engineering (ISSRE), 2016 IEEE 27th Int. Symposium on*, pages 207–218. IEEE, 2016.

[25] K. L. Ingham and H. Inoue. Comparing anomaly detection techniques for http. In *International Workshop on Recent Advances in Intrusion Detection*, pages 42–62. Springer, 2007.

[26] H.-P. Kriegel, A. Zimek, et al. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th Int. Conference on Knowledge discovery and data mining*, pages 444–452. ACM, 2008.

[27] R. Kwitt and U. Hofmann. Unsupervised anomaly detection in network traffic by means of robust pca. In *Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on*, pages 37–37. IEEE, 2007.

[28] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM Int. Conference on Data Mining*, pages 25–36. SIAM, 2003.

[29] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[30] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE Int. Conference on*, pages 413–422. IEEE, 2008.

[31] J. Mirkovic and P. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[32] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *Military Communications and Information Systems Conference (MilCIS), 2015*, pages 1–6. IEEE, 2015.

[33] N. Nostro, A. Bondavalli, and N. Silva. Adding security concerns to safety critical certification. In *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE Int. Symposium on*, pages 521–526. IEEE, 2014.

[34] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[35] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, volume 29, pages 427–438. ACM, 2000.

[36] S. Rosset and A. Inger. Kdd-cup 99: knowledge discovery in a charitable organization's donor database.

[37] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[38] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek. A framework for clustering uncertain data. *Proceedings of the VLDB Endowment*, 8(12):1976–1979, 2015.

[39] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.

[40] J. Song, H. Takakura, and Y. Okabe. Description of kyoto university benchmark data.

[41] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 535–548. Springer, 2002.

[42] F. J. Task. Security and privacy controls for federal information systems and organizations. *NIST Special Publication*, 800(53):8–13, 2013.

[43] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6. IEEE, 2009.

[44] T. Verwoerd and R. Hunt. Intrusion detection techniques and approaches. *Computer communications*, 25(15):1356–1365, 2002.

[45] X. Wan, W. Wang, J. Liu, and T. Tong. Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC medical research methodology*, 14(1):135, 2014.

[46] T. Zoppi, A. Ceccarelli, and A. Bondavalli. Exploring anomaly detection in systems of systems. In *Proceedings of the Symposium on Applied Computing*, pages 1139–1146. ACM, 2017.