Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

On the educated selection of unsupervised algorithms via attacks and anomaly classes

Tommaso Zoppi*, Andrea Ceccarelli, Lorenzo Salani, Andrea Bondavalli

Department of Mathematics and Informatics at University of Florence, Viale Morgagni 67/A, Florence, Italy

ARTICLE INFO

Keywords:
Anomaly detection
Intrusion detection
Attacks
Intrusion datasets
Unsupervised learning
Security
Data mining

ABSTRACT

Anomaly detection aims at finding patterns in data that do not conform to the expected behavior. It is largely adopted in intrusion detection systems, relying on unsupervised algorithms that have the potential to detect zero-day attacks; however, efficacy of algorithms varies depending on the observed system and the attacks. Selecting the algorithm that maximizes detection capability is a challenging task with no master key. This paper tackles the challenge above by devising and applying a methodology to identify relations between attack families, anomaly classes and algorithms. The implication is that an unknown attack belonging to a specific attack family is most likely to get observed by unsupervised algorithms that are particularly effective on such attack family. This paves the way to rules for the selection of algorithms based on the identification of attack families. The paper proposes and applies a methodology based on analytical and experimental investigations supported by a tool to i) identify which anomaly classes are most likely raised by the different attack families, ii) study suitability of anomaly detection algorithms to detect anomaly classes, iii) combine previous results to relate anomaly detection algorithms and attack families, and iv) define guidelines to select unsupervised algorithms for intrusion detection.

1. Introduction

Modern systems such as cyber-physical infrastructures, Systems-of-Systems or Cloud environments may be targeted by cyber-attacks, requiring attentive security countermeasures. *Intrusion Detectors* (IDs, [2,24]) were proposed to enhance security by analysing system data, aiming at identifying error-prone, malicious or unauthorised activities. IDs may apply signature-based techniques [2], which consist of checking properties or looking for patterns (*signatures*) in monitored data to detect the manifestation of a fault, or an ongoing attack.

Signature-based approaches have good detection capabilities when dealing with known faults or attacks [1,2], but they may fail in identifying unknown faults. In addition, when an unknown fault or a *zero-day attack* [28,29] (i.e., an attack that exploit novel or undiscovered system vulnerabilities) is revealed, a new signature must be promptly devised and added to the signatures set.

To deal with unknowns, research moved to techniques suited to detect unseen, novel attacks. Anomaly detectors are based on the assumption that an attack generates observable deviations from the expected behavior, and they aim at *finding patterns in data that do not conform to the expected behavior of a system* [1]: such patterns are known as *anomalies*. Once an expected behavior is defined, anomaly detectors target deviations from such expectations, protecting against known attacks [35,58] zero-day attacks [26,28] emerging threats [15,54] and enhancing existing algorithms [56]. In this paper we focus on *unsupervised anomaly detection algorithms, which are suited to detect, among others, zero-day attacks*, with no need of labels in training data [9,24].

Alongside with an appropriate quality of input data, selecting the correct detection algorithm(s) represents a key decision when defining an anomaly detector. Since IDs should be configured and customized to suit the target system, adequate strategies to guide the selection of appropriate anomaly detection algorithms may support and speedup the process. However, a clear support to this selection process in the domain of IDs is currently not available. The *scarcity of valid guidelines is mainly due to difficulties in extracting common characteristics of either attacks or algorithms*, which forces ad-hoc customizations.

Our contribution. This paper investigates which anomalies are generated when attacks occur, and determines which algorithms are more suited to detect specific anomaly classes. Consequently, *our study allows determining which algorithms are more suited to de-*

* Corresponding author.
E-mail addresses: tommaso.zoppi@unifi.it (T. Zoppi), andrea.ceccarelli@unifi.it (A. Ceccarelli), andrea.bondavalli@unifi.it (A. Bondavalli).

https://doi.org/10.1016/j.jisa.2020.102474

tect certain attacks. On top of that, the paper generalize this result, presenting the connection between attack families and anomaly classes: the implication is that it is possible to select anomaly detection algorithms that are particularly suited to detect attacks, even unknown ones, as long as these attacks belong to the same family. This ultimately provides *guidelines to identify the most suitable unsupervised algorithms for anomaly-based intrusion detection* for a given attack model.

We substantiate our results through a multi-level study, which devises initial conjectures that are then confirmed or denied by both inspection of data and by experimental campaigns. Briefly, we proceed as follows. First we report on anomaly classes and attack families according to reference taxonomies. This allows identifying which anomaly classes are generated when attacks occur. Such analysis is based on inspection of known attacks datasets typically used for testing IDs, along with appropriate experimental campaigns. Then, we define which algorithms are more suited to detect certain anomaly classes, exploring their characteristics and exercising them using databases in which we injected different anomaly classes. We select unsupervised algorithms belonging to different families [1,5,48], building a pool of algorithms with heterogeneous characteristics. Moreover, we favor well-known and consolidated algorithms with availability of public implementations rather than recent findings. This allows evaluating how the baseline idea behind algorithms belonging to different families suits the detection of specific classes of anomalies.

To corroborate the partial results obtained at the previous steps - and deriving guidelines - algorithms are connected to attack families, executing the selected algorithms on well-known intrusion datasets. Data generated or presented in the paper is publicly available [21], as well as the tool used for the experiments [36], allowing to reproduce experiments as needed.

Paper Structure. This paper is structured as follows: Section 2 presents basics and related works, while Section 3 describes the methodology we used throughout the paper. Section 4 expands on anomaly classes generated by attack families; Section 5 digresses on suitability of algorithms in detecting anomaly classes, while the studies above are consolidated in Section 6. Section 7 proposes guidelines to apply our study in IDs design, letting Section 8 to conclude the paper, elaborating on future works.

2. Basics and related works

2.1. Anomaly-Based intrusion detection

Aiming at protecting cyber-physical systems, security specialists are continuously researching mechanisms and strategies that aim at neutralizing an attack or mitigating its adverse effects. Regardless of their characteristics, attacks [26,27] should be timely identified to activate reaction mechanisms that specifically aim at blocking an ongoing attack, or protecting critical data.

To such extent, many IDs were proposed in the literature - and often distributed as enterprise software – to prevent attackers from exploiting security breaches, or vulnerabilities. Significant effort was put in comparing anomaly detection algorithms: for example, in [52] authors used 7 algorithms on a dataset containing HTTP traffic. Instead, in [35], authors presented a comparative study for IDs where *k-Nearest Neighbors* (kNN), *Mahalanobis*-based, *Local Outlier Factor* (LOF) and *one-class Support Vector Machines* (SVM) were evaluated using the DARPA 98 dataset. Four algorithms are evaluated in [51], which presents a review of novelty detection methods that are classified as semi-supervised or unsupervised. Additionally, in [5], authors presented a comparison of anomaly detection algorithms for multivariate data points. In this case, 19 algorithms were evaluated using 10 different datasets from different areas. Instead, in [48], authors focus on quantitative comparisons of unsupervised algorithms for intrusion detection, drawing conclusions about the effectiveness of algorithms on different datasets.

However, changing the target system, domain or attack model requires that the process of building the IDS to re-start from scratch. Therefore, in this paper we expand on the investigation on detection capabilities of algorithms, and on how different attacks generate anomaly classes, aiming at achieving general guidelines to select unsupervised algorithms when building IDs.

2.2. Unsupervised anomaly detection

In the paper we will name *data point* the observation of the state of the system at a given instant. Each data point is composed by f feature values, which are processed by an anomaly detection algorithm to determine if the data point exhibits anomalies. More in detail, anomalies are *rare data points* that may be classified as [1]:

- *point* anomaly (outlier): a data point that is out of scope or not compliant with the trend of a variable e.g., out-of-size payload of a network packet;
- *contextual* anomaly: a data point that is unexpected in a specific context e.g., low number of page faults while loading a program for the first time;
- *collective* anomaly: a collection of related data points that is anomalous with respect to the entire trend or dataset e.g., subsequent ICMP requests in a short interval of time.

Different anomaly detectors may be instantiated depending on the nature of the target system [1] and monitored data. If labeled training data is available, *supervised anomaly detection* [57,58,61] or *semi-supervised* may be adopted [25]. Labelled data points allow training an algorithm using both expected and anomalous data points that have already been reported. Another slightly different approach embraces ensembles [59,60], which are executed simultaneously, and the results they individually obtain are merged together to obtain the final class to be assigned to a give data point. It is worth noticing that feeding the algorithm with anomalies due to known attacks makes it learning how such attacks differ from expectations, disregarding the detection of anomalies due to unseen attacks. Instead, when training data is not available or labeled, the only option is an *unsupervised anomaly detection* approach [5,24].

Noticeably, when configuring an anomaly detector for a target system, we can assume that a fully labeled training set will not be available in most of the cases due to i) lack of trustable labeling techniques, ii) difficulties in gathering reliable data, or iii) dynamic and evolving characteristics of the system and its workload, calling for adaptive data analysis solutions. As a consequence, the applicability of supervised algorithms may not be guaranteed in several scenarios, calling the adoption of techniques that are able to deal with the scarcity of labels in training data. In addition, consolidated supervised algorithms may and should be used alongside with algorithms that are able to deal also with unknown attacks, building ensembles [59,60] or - more in general - IDSs that are able to identify a broader span of attacks. Therefore, in this paper *we refer only to unsupervised algorithms*.

2.3. Families of unsupervised algorithms

We describe here six families of unsupervised algorithms typically acknowledged in the literature [1,5], reporting their main characteristics. It is worth noticing that there are some unavoidable semantic overlaps among families. For example, neighbor-based strategies may be used to improve the detection capabilities of algorithms as in the angle-based FastABOD [4].

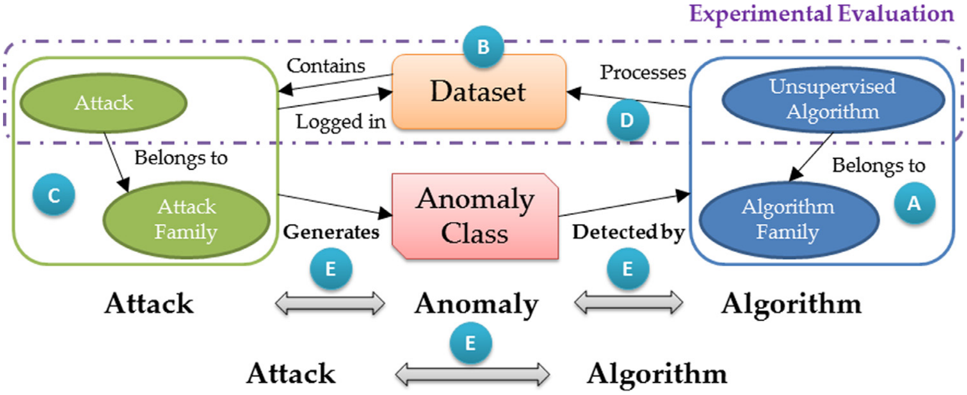


Fig. 1. Motivation of the Study, along with the methodology (steps A-E).

Clustering algorithms [11] partition a set of data points in such a way that data points in the same group (cluster) share similar characteristics. Data points that cannot be assigned to any of the existing clusters, or that do not met specific inclusion criteria, are anomalous.

Neighbor-based algorithms learn by analogy: they label a data point as anomalous or expected depending on the label of its nearest neighbor(s), considering an *f*-dimensional space [6,7]. In unsupervised mode, they use the distance of the data point from its neighbors as anomaly score.

Angle-based algorithms relate data to high-dimensional spaces, and measure the variance in the angles between the data point to the other points [4]. Expected data points have a large angle variance, while anomalies typically result in very small variance of triples of points.

Classification algorithms identify the class a new data point depending on information collected during previous activities e.g., assigning a given email into spam or non-spam classes. Despite they were born for supervised setups, they can run unsupervised [9].

Density-based algorithms [10] estimate the density of the neighborhood of each data point. When a data point differs from the expectations, it lies in a low-density area and it is then labeled as anomalous.

Statistical algorithms assume that expected data points occur in high probability regions of a given statistical distribution. They fit a distribution to the expected points, and then apply statistical inference to determine if a novel data point belongs to this distribution or not. In unsupervised mode, statistical algorithms [3] derive the underlying distribution as data is computed.

3. Methodology

This section lists the main steps of the experimental methodology we propose and apply in the paper. Such steps describe the selection of algorithms, datasets, attacks, metrics and tool support, providing the pillars to build our analysis by means of *qualitative and quantitative studies*. The former aims to define hypotheses and conjectures based on literature reviews or manual inspections, which are scrutinized by the latter through experimental campaigns. Our methodology is depicted in Fig. 1.

Step A. Alg*orithms Selection*. We select an unsupervised anomaly detection algorithm for each of the families above by surveying what researchers and practitioners proposed and in the literature. Algorithms must have been applied for intrusion detection in the past.

Step B. Dataset selection. Algorithms will be exercised on public datasets obtained by monitoring systems and reporting on effects of either real or simulated attacks on these systems. Datasets should be adequately documented, recent, or widely used in the past in similar studies.

Step C. Attacks *classification.* Attacks contained in the datasets should be classified according to a unified attack model, partitioning them into families according to their intrinsic characteristics.

Step D. Metrics *identification.* Then, we define metrics to score and compare the results of algorithms when applied to datasets. We select a reference metric, reporting also the values of metrics used in most of the other studies to allow state-of-the-art comparisons.

Step E. Exp*erimental Setup and Tool Support.* We select one or more tools that can support quantitative and, where needed, qualitative analyses. Similar analyses should be supported by the same tool, to limit variability of results e.g., the same implementation of an algorithm should be used. Moreover, the same environment should be defined and used to run experiments with tool support.

After defining the main steps of the methodology, a target of each study needs to be identified. Then, qualitative and quantitative analyses can be carried out, providing results to be used for discussion. Three separate studies will be carried out from Section 4 to Section 6. More in detail, Section 4 will report on *Attack Families That Generate Anomaly Classes*, Section 5 will expand on *Detecting Anomaly Classes*, while in Section 6 we will be *Applying Algorithms to Datasets*. As it is shown in Fig. 1, tool support (step E) is needed to conduct the three analyses.

3.1. Step A: algorithms selection

Our selection criteria favor well-known, consolidated algorithms with public implementations. Despite technical advancements that may have been proposed by domain experts, we assume that algorithms belonging to a given family share intrinsic strengths and weaknesses that may be mitigated, but that cannot be removed at all. Comparing consolidated versions of algorithms allow us to evaluate how the baseline idea behind the algorithms of a given family suits the detection of specific classes of anomalies. The selected algorithms, one for each family in Section 2.3, are described below.

Neighbor-based: ODIN. Stemming from the *k-th Nearest Neighbour* (kNN) [7], this distance-based method was designed to identify point anomalies. For each data point, kNN examines the whole dataset to determine their feature distances to the given point. This allows isolating *k* nearest neighbors (NN), creating the so-called *kNN graph*. The *Outlier Detection using Indegree Number* (ODIN, [6]) algorithm improves kNN by defining as anomalies the data points that have a low number of in-adjacent edges in the kNN graph.

Clustering: KMeans. K-means [11] assigns data points to *k* subsets, or *clusters*, by their feature values. First, *k* centroids are randomly initialized and each data point is assigned to the cluster with the nearest centroid. Centroids may be updated, fitting evolving scenarios also in unsupervised mode. Finally, data points that are too far from the centroid of their cluster are labeled as anomalies. There are many possible variants of this algorithm.

Angle-based: FastABOD. FastABOD anomalous data points depending on the angles between pairs of distance vectors to other points [4]. For each data point, the algorithm first calculates the *Angle Based Outlier Factor* (ABOF) to its k-nearest neighbor as the normalized scalar product of the difference vectors of any triple of neighbors. According to [4], the usage of kNN provides a better approximation. Then, *FastABOD* ranks the data points according to their ABOF. The smaller the ABOF, the bigger the probability that the data point represents an anomaly.

Classification: One-Class SVM. This algorithm conducts semi-supervised anomaly detection [1] aiming to learn a decision boundary [8]. However, *One-Class SVMs* can be used for unsupervised anomaly detection: a support vector machine is trained with the dataset and each data point is classified considering the normalized distance of the data point from the determined boundary [9].

Density-based: LOF. Local Outlier Factor (LOF) [10] computes the kNN for each data point, and use them to calculate the density index, called *Local Reachability Density* (LRD). The anomaly score is then obtained by comparing the LRD of a data point with the LRD of its kNN. Expected data points have scores close to 1.0, while anomalies usually result in bigger scores.

Statistical: HBOS. This approach [3] generates a histogram for each feature by using the values of all the available data points. The anomaly score is computed by multiplying the inverse heights of the columns in which each feature of the data point reside. Such technique assumes that the investigated features are independent, making HBOS fast even when dealing with large datasets [5]. If features are dependent, such dependencies need to be neglected.

Computational Complexity. Despite not central to our selection process, we report here some information on the computational complexity of the selected algorithms. The complexity of *density-based* and *angle-based* algorithms is at least $O(N^2)$, or rather the complexity of an NN query. *Clustering* is generally more efficient, with complexity of O(k N), where *k* represents the number of clusters. *Statistical* and *classification* families may have very different complexities; therefore it is not easy to present a bound. However, statistical algorithms are usually sub-linear e.g., O(N log N) for *HBOS*, while classifiers usually build complex structures or conduct complex calculations e.g., *OneClassSVM*, $O(N^2)$.

3.2. Step B: selection of the datasets

The datasets initial selection requires them to contain enough data points to ensure statistical evidence when evaluating the algorithms e.g., DARPA 1999 dataset [30] was discarded since it contains only 201 data points related to attacks, while ADFA-LD [55] contains just the number of system calls as usable feature. Furthermore, labels must be certain and not be assigned by classification algorithms or thresholds, to avoid biases due to mistakes in the labeling process. Consequently, we disregard datasets as MAWI [31] or DEFCON [33], which are constituted of sniffed data that is labeled applying classification algorithms. Last, data points should be complete for all the features in the datasets, to avoid applying feature recovery strategies that may bias results.

3.2.1. Selected datasets

The selected datasets are shortly described below and reported in Table 1. We match each dataset to an acronym that will be used throughout the paper. As a side note, during our selection process we discarded Kyoto2006+ [32] NGIDS-DS [33] and ADFA-LD [55] datasets, since their amount of data was too huge to be processed to a meaningful percentage.

Table 1
Datasets used in this study.

Dataset	Data Points	Attacks	%Attacks	Features
KDD Cup 99 (KC) [34]	311.028	223.298	72	41
NSL-KDD (NK) [16]	148.503	71.280	48	42
ISCX2012 (IX) [17]	571.698	66.813	12	17
UNSW-NB15 (UN) [14]	175.341	119.341	68	46

(KC) KDD Cup 99 (1999) [34]. This is the most popular dataset in the anomaly-based intrusion detection area, still used in recent experiments and surveys [5,18] and works prior the release of the updated NSL-KDD [20] despite being almost 20-years-old [55]. The dataset contains the following attacks: DoS (Denial of Service), R2L (unauthorised access from remote), U2R (unauthorised access to superuser/root functions) and Probing (gather network information).

(NK) NSL-KDD (2009) [16]. 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.

(IX) ISCX (2012) [17]. It is generated by the *Canadian Institute of CyberSecurity* 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 brute-force login attempts.

(UN) UNSW-NB15 (2015) [14]. This dataset was released by the *Australian Defense Force Academy, 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.

3.2.2. Synthetic datasets

To adequately support experiments in Section V, we also create sub-datasets of NK, IX and UN with respectively 48.084, 122.148, and 44.353 data points without attacks. We left KC out because most of its characteristics are shared with NK, especially when filtering out attacks. In addition, for each feature, we process sub-datasets to calculate statistical indexes e.g., *minimum* and *maximum* values, *average, median, variance*. These statistical indexes will be used to inject anomaly classes into the sub-datasets, according to their characterization in [1]. Despite injected anomalies do not perfectly replicate manifestations of real attacks, we simulate them to the best of our capabilities as follows:

1. Point anomalies: some feature values are updated with values that are either smaller than the minimum values or bigger than the maximum values logged in the dataset for a given feature.
2. Contextual anomalies: some feature values are updated with values that are not contained in the 95% confidence interval, considering the last 100 feature values as current context.
3. Collective anomalies: we inject a collective anomaly as a set of three subsequent data points where we updated some feature values with values outside the 80% confidence interval, considering the last 100 feature values as current context. Collective anomalies are not subsequent point or contextual anomalies; therefore we used a different confidence interval to generate them.

Table 2
Unified Attack Model and Mapping of Specific Attacks.

Attack Family	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

As a final step, we inject point, contextual and collective anomalies in the sub-datasets, obtaining a 95% - 5% ratio of expected – anomalous data points. The resulting sub-datasets NK-S, IX-S, UN-S are available [21].

3.3. Step C. unified attack model

Each of the datasets above uses inconsistent naming and grouping of attacks. To perform cross-datasets comparisons, we adopt the *unified attack model* that builds on [26,27,47] and is used in [46,48]. The model partition attacks in the following families: i) *communication passive*: attacks directed to gather or steal data through the passive observation of the communication channel, ii) *communication active*: attacks which use the communication channel as a way to send malicious data / requests to the target system, iii) *host*: malware or malicious code injected in a target host exploiting vulnerabilities of the operating system, and iv) *application*: attacks that exploits vulnerabilities of (web)services. The unified attack model is summarized in Table 2. The table reports also the mapping of all the different attacks referenced in the datasets to each of the 4 attack families. As example, *exploits* attacks of UNSW-NB15 (UN) fall into the application family, as it can be observed at the bottom right cell of the table. Attacks with different labels, or reported in different datasets, which resemble the same attack are merged into a unique attack, e.g., DoS, which can be found in both NSL-KDD (NK) and ISCX (IX) datasets.

3.4. Step D. Scoring Metrics

The effectiveness of anomaly detectors is usually scored through correct detections - *true positives* (TP), *true negatives* (TN) - and wrong detections (*false negatives*, FN), (*false positives*, FP). These indicators are commonly used to derive the so-called *confusion matrix*. Aggregate metrics based on the abovementioned indicators are *Precision, Recall* (or *Coverage*), *False Positive Rate, Accuracy, F-Score*(β) [18], *F-Measure* (F1), Area Under ROC Curve (*AUC*, [20]) and *Matthews Coefficient (MCC,* [19]). As highlighted in [49,53], under specific circumstances some metrics can be misleading, since they either i) do not consider all the four classes of the confusion matrix i.e., F1, FScore(β), or ii) consider all the classes without weighting the size of trues and falses i.e., Accuracy. To such extent, in this paper we mainly refer to *MCC* [19], which does not show the weaknesses above.

3.5. Step E. tool support and experimental setup

3.5.1. Tool support

To execute experiments, we need tools that allow i) executing the selected algorithms, and ii) extracting the metrics above. After examining well-known frameworks as ELKI [22], WEKA [23] or Pandas [50], our final choice has been RELOAD [36], an open-source tool that embeds implementations from different frameworks and runs experiments through a simple and intuitive user interface.

Briefly, RELOAD[1] automates the selection of the most relevant features out of a data set or data stream, which is typically very important in attack detection to reduce the amount of data to be observed. Further, it includes built-in metrics for the evaluation. The tool wraps the implementation of several unsupervised algorithms, amongst those which are often deemed the most useful [24,35] for unsupervised anomaly detection in cyber-security. Additionally, it embeds automatic tuning of algorithms' parameters, and facilitates examining outputs through reports of CSV files and graphical plots.

RELOAD also considers different *decision functions* to convert algorithms' scores into boolean e.g., anomaly, normal, scores: out of the pool of available ones, we chose *IQR* and *Confidence Interval* [39], which RELOAD shapes depending on the characteristics of the algorithm.

3.5.2. Experimental setup

We describe here the experimental setup we used throughout the paper. Starting from the data sets, we downloaded the source files of *KC, NK, IX* and *UN* from their repositories and we preprocessed them to shape such data as csv files, which can be efficiently processed by RELOAD. Then, we downloaded the latest release of RELOAD from the GitHub repository, setting up its parameters. We set MCC as target metric, using as feature selection strategies the *variance* of feature values and their *information gain* [38] with respect to the label. We also proceed with a 10-fold sampling of the training set as widely suggested [37] in the literature.

We executed different experimental campaigns including all the algorithms considered in this study. Parameters tuning is adopted by RELOAD to find an adequate setup of each instance of a given algorithm. Tuning is performed by i) first, executing training trying different combinations of parameters; ii) then, comparing results for the different parameters. For example, we run kNN-dependent algorithms i.e., *ODIN, FastABOD*, with $k \in \{1, 2, 3, 5, 10, 20, 50, 100\}$. In addition, metrics other than MCC e.g., *TP, FP, TN, FN, Precision, Recall, False Positive Rate, Accuracy* and *Area Under ROC Curve*, are reported for completeness and for easiness of comparison with existing studies.

Machine to Execute Experiments. The experiments have been executed on a server equipped with Intel Core i7-6700 with four 3.40 GHz cores, 24GB of RAM and 100GB of user storage. Overall, executing all the experiments supporting the quantitative analyses reported in this paper required approximately three weeks of 24H execution. We choose the portions considering the biggest subset of the dataset that do not escalate in heap memory errors, i.e., 16% for KC, 33% for NK, 20% for IX, 24% for UN and all the NK-S, IX-S, UN-S synthetic datasets. All metric scores, RELOAD data logs and files we used to collect and summarize values are publicly available [21].

4. Attack families that generate anomaly classes

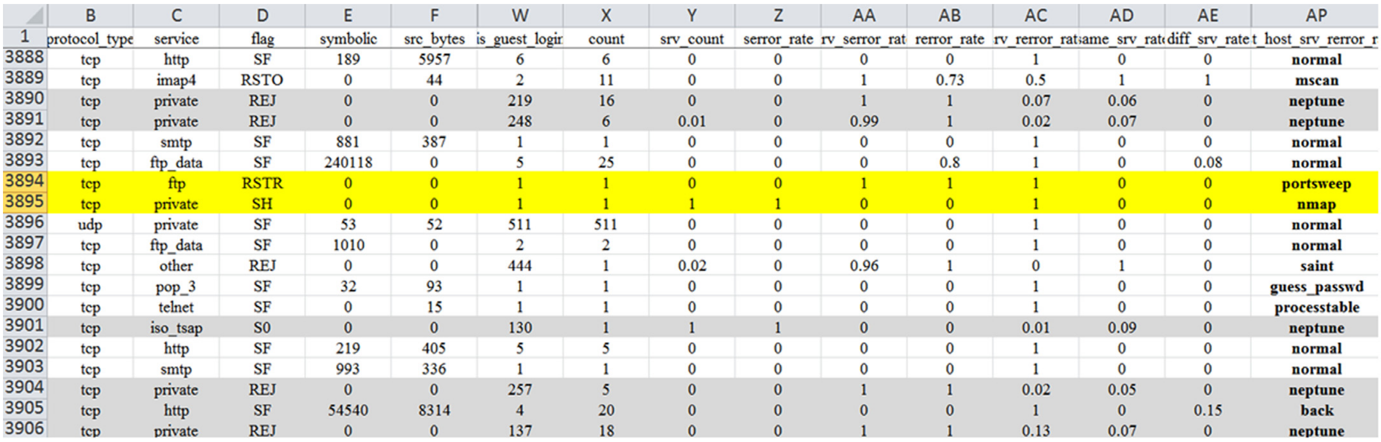
This section expands on qualitative and quantitative analysis directed to *identify the anomaly classes generated by the attacks contained in the datasets IX, KD, NK, UN* considered in the paper.

4.1. Inspection on the selected datasets

4.1.1. Attacks characteristics

We first consider the characteristics of attacks to identify how they usually manifest. Attacks as *Shellcode* or *Exploits* (appearing in

[1] RELOAD, GitHub Wiki, github.com/tommyippoz/RELOAD/wiki.



	B	C	D	E	F	W	X	Y	Z	AA	AB	AC	AD	AE	AP
1	protocol_type	service	flag	symbolic	src_bytes	is_guest_logir	count	srv_count	serror_rate	rv_serror_rat	rerror_rate	rv_rerror_rat	same_srv_rat	diff_srv_rate	t_host_srv_rerror_r
3888	tcp	http	SF	189	5957	6	6	0	0	0	0	1	0	0	**normal**
3889	tcp	imap4	RSTO	0	44	2	11	0	0	1	0.73	0.5	1	1	**mscan**
3890	tcp	private	REJ	0	0	219	16	0	0	1	1	0.07	0.06	0	**neptune**
3891	tcp	private	REJ	0	0	248	6	0.01	0	0.99	1	0.02	0.07	0	**neptune**
3892	tcp	smtp	SF	881	387	1	1	0	0	0	0	1	0	0	**normal**
3893	tcp	ftp_data	SF	240118	0	5	25	0	0	0	0.8	1	0	0.08	**normal**
3894	tcp	ftp	RSTR	0	0	1	1	0	0	1	1	1	0	0	**portsweep**
3895	tcp	private	SH	0	0	1	1	1	1	0	0	1	0	0	**nmap**
3896	udp	private	SF	53	52	511	511	0	0	0	0	1	0	0	**normal**
3897	tcp	ftp_data	SF	1010	0	2	2	0	0	0	0	1	0	0	**normal**
3898	tcp	other	REJ	0	0	444	1	0.02	0	0.96	1	0	1	0	**saint**
3899	tcp	pop_3	SF	32	93	1	1	0	0	0	0	1	0	0	**guess_passwd**
3900	tcp	telnet	SF	0	15	1	1	0	0	0	0	1	0	0	**processtable**
3901	tcp	iso_tsap	S0	0	0	130	1	1	1	0	0	0.01	0.09	0	**neptune**
3902	tcp	http	SF	219	405	5	5	0	0	0	0	1	0	0	**normal**
3903	tcp	smtp	SF	993	336	1	1	0	0	0	0	1	0	0	**normal**
3904	tcp	private	REJ	0	0	257	5	0	0	1	1	0.02	0.05	0	**neptune**
3905	tcp	http	SF	54540	8314	4	20	0	0	0	0	1	0	0.15	**back**
3906	tcp	private	REJ	0	0	137	18	0	0	1	1	0.13	0.07	0	**neptune**

Fig. 2. Example of collective anomalies: subsequent occurrence of probe attacks portsweep and nmap – in yellow – and DoS attacks neptune and back – gray rows - in NK dataset (Test sub-dataset, rows 3889 - 3906).

the UN dataset) aim at damaging a system with single requests or actions: therefore, we expect them to generate either point or contextual anomalies. Instead, attacks that submit several requests to a victim will generate separate anomalies that share common characteristics i.e., collective anomalies. Differentiating between attacks generating point anomalies or contextual anomalies is not trivial: in particular, it requires understanding if attacks act only in specific scenarios (context), and how they impact the victim.

More in detail, *Probing, Reconnaissance*, and *Analysis* attacks (e.g., *PortScan* [42]) aim at scanning system interfaces or network devices in order to determine vulnerabilities. This activity may be leveraged through time, sending multiple *ping* or *arp* requests that do not appear during normal operation conditions, generating collective anomalies. Other attacks as *Fuzzers* and *Bruteforce* aim at submitting inputs to a system to either block it or gain unauthorised access. Similarly, DoS attacks [26] aim to damage the victim leading to its unavailability. As example, denial of service may be exercised by sending many requests (also from different adversaries networked together, as in Distributed DoS, *DDoS*) or sending malformed packets [40], generating either collective or point anomalies.

Instead, we found that attacks belonging to the *Host* family in Table 2 generate contextual anomalies as follows. These attacks exploit system vulnerabilities to execute scripts (as it is for *Shellcode* or *Backdoors*), software (*Worms, Malware*), or change/obtain permissions (*r2L*, *u2r*) by directly executing instructions on the victim machine. Such attacks need to damage the system as quickly as possible, also hiding their activities (as it is common for *Camouflaged Worms* [41]), to avoid being detected and quarantined by antiviruses. Our conjecture is that they generate contextual anomalies, impacting the system in a short timespan without relevant fluctuations of feature values. For example, a worm may aim at scanning the system for passwords and sending them to some remote repository, or to change credentials for VPN or SSH access on a machine: sending data or changing permissions are not anomalies by themselves, but they are anomalous in specific contexts.

4.1.2. Manual inspections into datasets

Similarly to what was done in the previous section for algorithms, we verify our conjectures by examining how attacks manifest in the NK, KD, IX, and UN datasets. Manual inspections allow distinguishing which attacks appear as groups of anomalous data points rather than attacks that affect single data points. Fig. 2 highlights a section of the NK *Test* (note that NK is provided as two separate CSV files, one for train and one for test) dataset, where we can observe specific probing attacks as *ipsweep, nmap, portsweep*, or *satan*.

Subsequent data points – rows in the dataset – are labelled with one of the attacks above i.e., last column of rows 3894 and 3895 in Fig. 2, either *portsweep* and *nmap*. For DoS attacks, instead, datasets report both on single and groups of anomalies. Fig. 2 shows single (row 3901) and multiple occurrences of *neptune* attack (rows 3890, 3891), and also multiple occurrences of different DoS attacks i.e., *neptune* and *back*, rows 3904 – 3906.

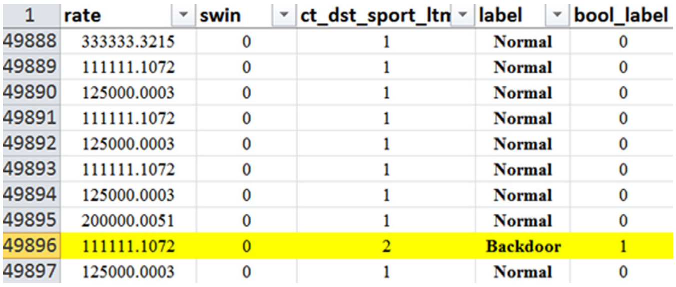
4.1.3. Deciding on point, contextual or collective anomalies

If an attack always appears in subsequent data points, we can definitively claim that such attack manifests as a collective anomaly. Otherwise, examining the datasets to differentiate between point and contextual anomalies requires more effort. We proceed as follows. We select the most relevant features of each of the datasets by calculating *Pearson correlation indexes* between each feature of the datasets and the label column, considering the label as 0 if normal, 1 otherwise. Then, we select the 3 features that are more relevant: the bigger the absolute value of Pearson index, the greater the correlation with the label. Moreover, we rank all the data points in the dataset according to these feature values and we carefully examine the results: if most of the attacks are either at the beginning or at the end of the ranked dataset, this means that attacks lead one or more of these features to show values outside the usual range, and therefore represent point anomalies.

In any other case, we check if such anomalies are contextual. Values related to the three features are processed to extract statistical indexes as average, median and standard deviation, differentiating between normal and anomalous data points. A noticeable difference between the two series of statistical indexes can suggest possible alterations due to this specific attack. Then, we select 5 portions of the dataset in which the attack appears. Once we find a row with the attack we look at the context in which the attack is put (i.e., up to 30 rows before and up to 5 rows later) to understand if, for some features, the data point corresponding to the attack has values that differ with respect to the "surrounding" data points, according to the statistical indexes calculated before.

4.1.4. Example: backdoor attack

To prove how our manual inspection works, we show as example the *backdoor* attacks, which we suspected to generate contextual anomalies. To confirm or deny the initial hypothesis, we first look for datasets containing this attack (UN, see Fig. 3), observing



1	rate	swin	ct_dst_sport_ltn	label	bool_label
49888	333333.3215	0	1	Normal	0
49889	111111.1072	0	1	Normal	0
49890	125000.0003	0	1	Normal	0
49891	111111.1072	0	1	Normal	0
49892	125000.0003	0	1	Normal	0
49893	111111.1072	0	1	Normal	0
49894	125000.0003	0	1	Normal	0
49895	200000.0051	0	1	Normal	0
49896	111111.1072	0	2	Backdoor	1
49897	125000.0003	0	1	Normal	0

Fig. 3. Example of contextual anomaly: occurrence of backdoor attack – in yellow – in UN dataset (rows 49,888 - 49,897).

that data points labeled as backdoor attacks mainly come individually, excluding collective anomalies. We then execute the feature selection process using *pearson index*. Out of the possible 46 features, the 3 that show higher correlation are *rate* (0.249 of pearson index), *swin* (−0.247) and *ct_dst_sport_ltm* (0.357). Ranking the dataset according to the values of these features does not show attacks that are concentrated either at the beginning or at the end: therefore, we cannot classify backdoor attacks as point anomalies. We then compute average, median and standard deviation for each of the three features, differentiating normal and anomalous data points. In particular, we notice how *avg* ± *std* values for *ct_dst_sport_ltm* are separated between normal (1.388 ± 0.847) and anomalous (2.234 ± 0.656) values, indicating average higher values of this feature in presence of attacks. Most of the data points labeled as backdoor attacks in the dataset show different values with respect to their context. As example, row 49,896 in Fig. 3 is the only data point that has value 2 for the *ct_dst_sport_ltm* feature considering rows 48,866 – 49,900 (30 rows before the attack – 5 rows after the attack).

Manual inspections, as shown above, are executed for each attack that is logged in the considered datasets, while final results are summarized in Table 3 and Table 4. The tables report on anomaly classes generated by i) attacks (Table 3), and ii) attack families (Table 4).

4.2. Experimental analysis

We now proceed to a dedicated quantitative analysis to confirm or deny manual inspections. Despite unsupervised algorithms execute training without relying on labels in the data, a tuning phase is employed by RELOAD to derive the optimal values of the parameters (e.g., the size *k* of the neighbourhood for *kNN*, when needed) for each algorithm. Tuning requires extracting a subset of the dataset – the *tuning subset* - containing expected data points and data points collected while a given attack – the *tuning attack* – was exercised, computing metric scores for each parameters' combinations.

Table 3
Attacks and Anomaly Classes they Generate.

Attacks	Datasets	Anomalies
Denial of Service	KC, NK, IX, UN	Point, Collective
Distributed Denial of Service	IX	Contextual, Collective
Probing / Reconnaissance	KC, NK, UN	Collective
R2L, U2R	KC, NK	Contextual
Infiltration, Bruteforce	IX	Collective
Shellcode, Backdoors, Worms	UN	Contextual
Fuzzers, Analysis	UN	Collective
Exploits	UN	Point

Table 4
Attack Families and Anomaly Classes they Generate.

Family	Attacks	Anomalies
Communication - Passive	Probing, Infiltration, Reconnaissance, Analysis	Collective

Once the training-tuning phases are completed, algorithms are ready to provide anomaly scores for data points in the *evaluation set*. Using an evaluation set that contains a *target attack* other than the tuning attack allows to completely decoupling the training-tuning phases from the evaluation, leading algorithms to detect unknown attacks. Indeed, *we expect to have higher detection scores in detecting unknown target attacks that generate the same anomaly classes of the tuning attacks contained in the tuning subsets*. Therefore, we proceed as follows:

- For each dataset we choose the target attack, identifying an evaluation set containing both normal data points and data points collected when the target attack was exercised.
- We then identify different tuning subsets of the dataset containing normal data points and data points related to an attack (i.e., the tuning attack), other than the target attack.
- Finally, we run experiments by training anomaly detection algorithms with normal data points, tuning them using a tuning subset, and then using trained algorithms to score the evaluation set.

4.3. Results and discussion

Table 5 reports on the analysis regarding three different target attacks: *bruteforce* (ISCX dataset), *u2r* (NSL-KDD), and *worms* (UNSW). For the full list of experiments please refer to the files available at [21]. For each target attack we report on the average and standard deviation scores we obtained by training, tuning and then evaluating the 6 algorithms we used in this study. In addition, each row reports on i) the dataset used, ii) the target attack and the anomalies we suppose it generates, Ii) the tuning attack and the anomalies we suppose it generates.

As it can be noticed in the table, highest *MCC* scores are obtained when *tuning attack* corresponds to *target attack*. More importantly, we point out that *when tuning attack and target attack differ, the highest scores are obtained when the anomalies generated by the tuning attack match the anomalies generated by the target attack*. For example, *Worms* attacks (see the last batch of rows in Table 5) are on average identified with better scores by algorithms if tuning is carried out using an tuning subset containing attacks generating contextual anomalies. The same trend can be observed also for *U2R* and *Bruteforce* attacks, as reported in rows 4–7 and 8–11 of Table 5. The results of this experimental campaign allow consolidating the link between attacks and anomaly classes that was figured out, as a first stage, through manual inspections.

5. Detecting anomaly classes

We discuss here *how the characteristics of each algorithm impact their ability in identifying specific anomaly classes*. Discussion is carried out by revising algorithms as they were proposed, and then corroborating initial conjectures with quantitative analyses.

5.1. Investigating characteristics of algorithms

Each unsupervised anomaly detection algorithm relies on its own properties. According to their characteristics [4,7,10], algorithms belonging to *neighbor-based, angle-based* and *density-based* families such as *ODIN, FastABOD* and *LOF* are primarily intended to identify point anomalies, while they may not be able to detect collective anomalies if the size *k* of the neighborhood is smaller than the size of the collective group of anomalies. Indeed, an indegree score may help detecting them, as demonstrated in [6]. This

Table 5
Detecting Unknown Attacks by Combining different Tuning Attack and Target Attack.

Dataset	Target Attack		Tuning Attack		MCC	
	Name	Anomaly	Name	Anomaly	avg	std
IX	*Bruteforce*	*Collective*	*Bruteforce*	*Collective*	0.82	0.17
IX	Bruteforce	Collective	Infiltrator	Collective	0.31	0.21
IX	Bruteforce	Collective	DoS	Collective (Point)	0.27	0.09
NK	*U2R*	*Contextual*	*U2R*	*Contextual*	0.35	0.07
NK	U2R	Contextual	R2L	Contextual	0.19	0.08
NK	U2R	Contextual	Probe	Collective	0.05	0.04

issue is shared also with some implementations of *KMeans* (a *clustering* algorithm), since a collective anomaly may lead them to create a separate cluster for the particular group of data points. However, variants as [12] mitigate this problem by labeling as anomalous both data points i) belonging to small clusters and ii) far from known clusters. *Classification* algorithms for unsupervised anomaly detection are often identified as *One-Class SVM* [9]. It is worth remarking that algorithms selected for this study – HBOS, KMeans, FastABOD, LOF, ODIN and OneClass SVM – are intended to represent the intrinsic characteristics of different families, rather than the most recent variants of existing algorithms.

5.2. Experimental analysis

Starting from the NK-S, IX-S and UN-S reported in Table 1, non-numeric features were removed, and feature selection was executed by RELOAD, leaving 16 features for NK-S, 5 for IX-S, and 13 for UN-S. During the training phase, RELOAD respectively creates 38, 20, and 25 feature subsets according to *Pearson Correlation*. This results in 42 feature subsets for NK-S, 26 for IX-S, and 39 for UN-S, which sum i) the single features, ii) the connected features, and iii) the set composed by all the single selected features. All the instances are evaluated together during training phase and all the possible algorithm configurations are ranked according to MCC.

Fig. 4 shows the metric scores we obtain by running the selected algorithms on our three synthetic datasets, and aggregating scores by algorithm and anomaly classes. Six series of three columns can be found in the figure: the columns on the left report on the detection of point anomalies, central columns report on contextual anomalies, while columns on the right describe the detection of collective anomalies. Such figure allows observing that SVM shows the highest average scores, while sub-quadratic algorithms as HBOS and K-Means show overall worse results i.e., bars are shorter in Fig. 4. However, we can observe how algorithms with lower bars may be really effective in detecting specific anomaly classes, i.e., *KMeans* and *FastABOD*. Point anomalies are detected well by *FastABOD* (see columns on the left of each triple of columns in Fig. 4), while *KMeans* show good capabilities in identifying collective anomalies (columns on the right of the triples in Fig. 4). *ODIN* and, to a lesser extent, *LOF*, are balanced algorithms: they do not show specific weaknesses for any anomaly class. Lastly, we remark how *HBOS*, despite not optimal in terms of metric scores, is a light and fast algorithm, making it useful when a computational resources are scarce.

Overall, we expected point anomalies to be the easiest to detect. Instead, except for the angle-based *FastABOD*, it was easier to detect collective anomalies than contextual and point anomalies. We explain this result as follows: the selected algorithms use large training sets, which allow a careful and precise definition of the boundaries between anomalous and expected behavior. However, during training they derive a global boundary, which does not always fit the detection of single anomalies, while groups of anomalous data points become easier to identify.

5.3. Results and discussion

Overall, our results show which unsupervised algorithms are more suitable than others to detect anomalies belonging to specific classes. The final results of our analyses are depicted in Fig. 5, which is built considering algorithms as capable of identifying an anomaly class if in our experiments the average *MCC* obtained by the algorithm exceeds 0.6. We choose this MCC threshold by considering each algorithm as capable of identifying at least an anomaly class.

We observe how Fig. 5 shows a few mismatches with respect to our conjectures from Section 5.1. We supposed *clustering* algorithms to be capable of identifying both point and collective anomalies, while Fig. 5 puts *KMeans*, the clustering algorithm, in the intersection of “contextual” and “collective” sets. Instead, SVM scores confirm that under average conditions i.e., the boundary of SVM is found, this algorithms is the most balanced overall. Moreover, as reported previously, higher scores are obtained by the algorithms that have quadratic computational complexity while

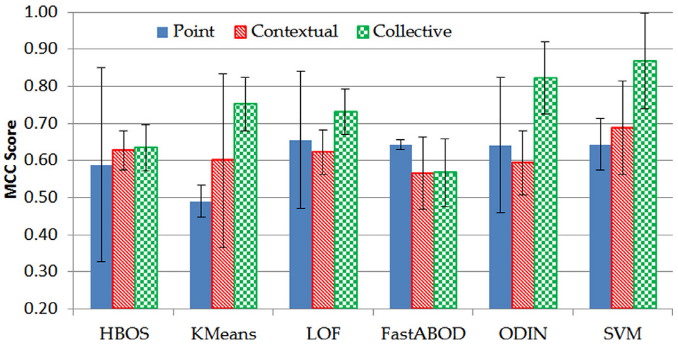


Fig. 4. Average metric scores of algorithms for each class of anomalies. Er-ror bars represent standard deviation among the datasets NK-S, IX-S, UN-S.

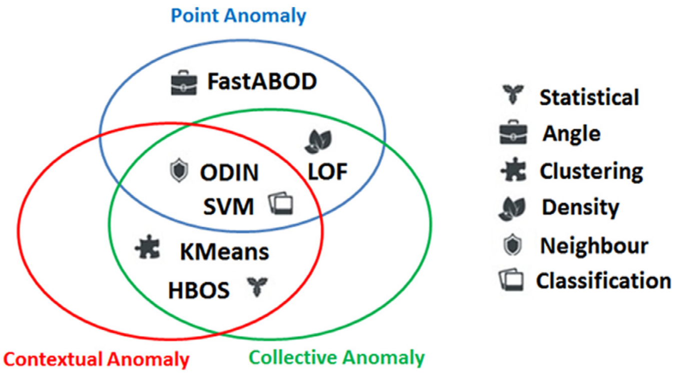


Fig. 5. Linking anomaly classes to algorithms depending on quantitative analyses by executing algorithms on datasets.

Table 6
Metric Scores obtained by applying each algorithm on the datasets KD, NK, IX, UN. Results are grouped by Algorithm and Anomaly Class Generated by the Attacks Evaluated through Algorithms.

Algorithm	Anomaly	TP	TN	FP	FN	FPR	P	R	F1	ACC	AUC	MCC (Fig. 6)		Expected MCC (Fig. 4)		Diff wrt Expected
		%	%	%	%							avg	std	avg	std	
HBOS	Point	21.98	58.44	10.88	8.70	0.15	0.61	0.72	0.59	0.80	0.79	0.49	0.26	0.59	0.26	19.7%
	Contextual	5.75	86.26	5.10	2.89	0.06	0.52	0.54	0.50	0.92	0.74	0.47	0.29	0.63	0.05	32.1%
	Collective	5.34	81.24	5.95	7.47	0.07	0.49	0.52	0.46	0.87	0.73	0.41	0.09	0.63	0.06	55.0%
K-Means	Point	23.89	57.29	12.03	6.79	0.17	0.57	0.63	0.58	0.81	0.73	0.45	0.31	0.49	0.04	7.7%
	Contextual	5.88	85.11	6.25	2.76	0.08	0.54	0.59	0.47	0.91	0.76	0.47	0.21	0.60	0.23	27.4%
	Collective	7.97	83.04	4.15	4.83	0.05	0.66	0.58	0.58	0.91	0.77	0.56	0.21	0.75	0.07	35.4%
LOF	Point	19.93	62.62	6.70	10.76	0.11	0.66	0.71	0.60	0.83	0.80	0.50	0.14	0.66	0.18	29.8%
	Contextual	6.77	84.45	6.90	1.88	0.08	0.50	0.66	0.53	0.91	0.79	0.51	0.17	0.62	0.06	22.5%
	Collective	7.95	82.81	4.38	4.85	0.05	0.62	0.65	0.61	0.91	0.80	0.57	0.18	0.73	0.06	28.7%

faster algorithms as *KMeans* and *HBOS* showed worse detection capabilities.

6. Applying algorithms to datasets

We *now apply the selected algorithms to datasets, analysing metric scores* to understand if they confirm or deny the results obtained in Section 4 and Section 5. We skip qualitative analyses, going directly to experimental evaluations.

6.1. Experimental analysis

We set an experimental campaign using the real datasets KD, NK, IX, and UN. We executed an experimental campaign by applying the 6 algorithms on the datasets KD, NK, IX and UN, according to the procedure described in Section 4. In Fig. 6 we report results of this experimental campaign, while the detailed metric scores are in Table 6. The results herein were obtained by applying the algorithms on the datasets, then grouping by attacks and, ultimately, by anomalies through the analysis in Section 4.

6.2. Results and discussion

Focusing on Fig. 6, we can observe that bars are lower then their counterparts in Fig. 4. This is pointed out in the last column of Table 6 and is explained considering that synthetic datasets were artificially created by injecting anomalies with a common pattern, making their detection easier than real attacks.

However, it is worth noticing that the characteristics of algorithms are confirmed. Algorithms that were classified in Fig. 5 to be particularly effective in detecting specific anomaly classes show metric scores that comply with our expectations. LOF, ODIN and SVM (respectively, *density-based*, *neighbor-based* and *classification*) have good overall capabilities for all anomaly classes. Fast algorithms as HBOS (*statistical*) and K-Means (*clustering*), instead, show results that are on average worse than algorithms with quadratic complexity. Moreover, HBOS shows bad scores in identifying collective anomalies, while it was supposed to detect collective anomalies quite well in Section 5.

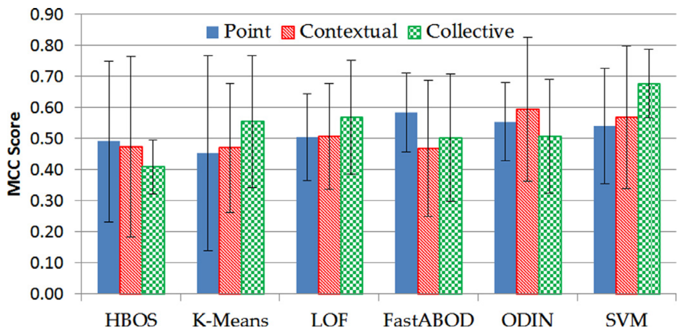


Fig. 6. Average metric scores of algorithms for each anomaly class. Error bars represent standard deviation among datasets KD, NK, IX, UN.

While all the algorithms show worse performance with respect to the results on synthetic datasets (24.8% on average), *FastABOD* shows scores that are similar to the analysis on synthetic datasets (14.4% worse on average), achieving the best average score in detecting attacks generating point anomalies.

7. On the design of anomaly-based IDs

The work described in the previous Sections allows establishing links between i) attacks – or attack families – and anomaly classes they usually generate (Section 4), and ii) algorithms and anomaly classes depending on detection capabilities (Section 5). These two separate studies – consolidated in Section 6 - constitute the *baseline to derive the most suitable unsupervised anomaly detection algorithm(s) for intrusion detection in a given system*. This ultimately allows building guidelines to identify the most suitable anomaly detection algorithms for a given system, assuming that the system has enough computational, memory and storage resources to support the algorithms execution. *Here we have considered a representative set of algorithms and dataset, however our guidelines may be generalized for a selection of different algorithms, dataset and attacks.*

Guidelines are summarized in Table 7, where we give a brief description of possible attack families that can impact a system, pointing out the anomaly classes they usually generate, and then proposing algorithm(s) to apply for intrusion detection. More in detail, Table 7 reports on all the 7 possible combinations of point, contextual and collective anomaly classes. The system administrator, the researcher or the practitioner that wants to take advantage of our design guidelines has to first devise a list of possible attacks that may threaten the system, building an attack model e.g., *DoS and probing attacks*. Then, each attack of the attack model should be shaped according to the attack families we presented in Table 2 e.g., *communication – active and communication – passive families*. Once this model is done, attack families derived from the attack model should be used to find a matching item in the first column of Table 7 (e.g., the third row of the table for communication attacks). In this way, *the user can get immediate recommendations about the most suitable algorithm(s) to adopt to detect intrusions as defined in the attack model.*

When devising an attack model is not possible or feasible, the researcher or practitioner may want to be covered against a wide range of attacks. To such extent, the last row of Table 7 reports on algorithms that showed overall good capabilities in detecting all the three anomaly classes, meaning that they offer a balanced coverage against most of the possible manifestations of attacks, either known or unknown.

Table 7
Guidelines for Designing Intrusion Detectors based on Anomaly Classes.

Attack Families	Anomaly Classes	Algorithms (Families)	Motivation
Application	Point	ODIN (Neighbour), FastABOD (Angle)	They show good scores in detecting point anomalies. We may consider to use both strategies in parallel, since the core of ODIN and FastABOD share a kNN search.
Host	Contextual	ODIN (Neighbour), SVM (Classification)	SVM is a better choice if additional coverage against attacks generating collective anomalies is desired. Otherwise, SVMs fit the most here.
Communication - Passive, Communication - Active	Collective	SVM (Classification), KMeans (Clustering)	SVM shows the best detection capabilities in detecting collective anomalies. Clustering may be a good alternative if the system cannot rely on fast CPU or enough RAM memory
Host, Application	Point, Contextual	FastABOD (Angle), ODIN (Neighbour)	They show good capabilities in detecting both anomaly classes.
Host, Communication - Passive	Contextual, Collective	SVM (Classification)	It shows good capabilities in detecting both anomaly classes.

Finally, in our analysis, we observe that all algorithms but *HBOS* and *KMeans* rely on actions e.g., finding the k-NNs or creating graphs, that are quadratic with respect to the amount of the data points used for training. To mitigate this problem, in some scenarios as the detection of attacks generating collective anomalies (see third row of Table 7), it is possible to adopt algorithms that have slightly worse detection capabilities, but on the other side require considerably less resources for their training. Other possible solutions could rely on sliding windows algorithms [12,13,43], which depend on a fixed size of data points to train their model, limiting the effort required during training phase. However, despite showing promising results in some application domains [45], to the best of our knowledge such strategies are not mature enough [44] for being compared with more consolidated algorithms for detailed analyses.

8. Conclusions

In this paper we described activities on attacks, anomalies and unsupervised algorithms, which allowed deriving guidelines to select unsupervised anomaly detection algorithms for intrusion detection. Guidelines constitute the last step of a process which allowed: i) to characterize known attacks in terms of the anomaly classes they usually generate, i) to study suitability of anomaly detection algorithms to detect anomaly classes, iii) to establish a link between anomaly detection algorithms and attacks through anomaly classes, iv) and to take advantage of the link to propose guidelines to select unsupervised algorithms. Steps i) – iii) started from qualitative analyses then substantiated by quantitative ones using state-of-the-art algorithms, attack models and datasets. All data generated for these quantitative analyses is publicly available [21], as well as the tool used for the experiments [36].

Overall, such analyses escalated into guidelines to derive the most suitable unsupervised anomaly detection algorithm for intrusion detection in a given system depending on anomaly classes. In addition, we carefully described the methodology we followed, to promote further expansions of our work e.g., by considering unsupervised algorithms other than the 6 we took into account. Our guidelines could also be used to suggest a set of unsupervised algorithms: however, to the best of our knowledge, there are no consolidated mechanisms that allow combining individual anomaly scores of different algorithms in an efficient and trusted manner.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tommaso Zoppi: Conceptualization, Methodology, Writing - original draft, Investigation, Software, Validation, Writing - review & editing. **Andrea Ceccarelli:** Conceptualization, Methodology, Writing - original draft, Visualization, Validation, Supervision, Writing - review & editing. **Lorenzo Salani:** Data curation, Investigation, Methodology, Validation. **Andrea Bondavalli:** Visualization, Supervision, Writing - review & editing, Project administration.

Acknowledgement

This work has been partially supported by the REGIONE TOSCANA POR FESR 2014–2020 SISTER and by the H2020 program under the Marie Sklodowska-Curie grant agreement 823788 (ADVANCE) projects.

References

[1] Chandola, V., Banerjee, A., Kumar, V. “Anomaly detection: a survey”. (2009) ACM computing surveys (CSUR), 41(3), 15.
[2] Modi Chirag, et al. A survey of intrusion detection techniques in cloud. Journal of Network and Computer Appl 2013;36(1):42–57.
[3] Goldstein Markus, Dengel Andreas. "Histogram-based outlier score (hbos): a fast unsupervised anomaly detection algorithm. KI-2012: Poster and Demo Track 2012:59–63.
[4] Kriegel H.-.P., Zimek A. “Angle-based outlier detection in high-dimensional data”. Proc. of the 14th ACM SIGKDD Int. Conference on Knowledge discovery and data mining; ‘08. p. 444– 52.
[5] Goldstein M, Uchida S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS ONE 2016;11(4):152–73 p.e.
[6] Hautamaki V, Karkkainen I, Franti P. Outlier detection using k-nearest neighbour graph. in pattern recognition. ICPR 2004. Proceedings of the 17th International Conference on 2004;3:430–3 August. IEEE.
[7] Liao Y, Vemuri VR. Use of k-nearest neighbor classifier for intrusion detection. Computers&Security 2002;21(5):439–48.
[8] Scholkopf B, Platt JC, Shawe-Taylor J, Smola AJ, CWilliamson R. Estimating the support of a high-dimensional distribution. Neural Comput 2001;13(7):1443–71.
[9] Amer M, Goldstein M, Abdennadher S. Enhancing one-class support vector machines for unsupervised anomaly detection. in Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description. ACM 2013:8–15.
[10] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: identifying density-based local outliers. ACM sigmod record 2000;29(2):93–104 May. ACM.
[11] Schubert E, Koos A, Emrich T, Züfle A, Schmid KA, Zimek A. A framework for clustering uncertain data. Proceedings of the VLDB Endowment 2015;8(12):1976–9.
[12] Zhou A, Cao F, Qian W, Jin C. Tracking clusters in evolving data streams over sliding windows. Knowl Inf Syst 2008;15(2):181–214.
[13] Zhang Liangwei, Lin Jing, Karim Ramin. "Sliding window-based fault detection from high-dimensional data streams. IEEE Transactions on Systems, Man, and Cybernetics: Systems 2017;47(2):289–303.
[14] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). Military Communications and Information Systems Conference (Mil-CIS) 2015:1–6 IEEE, 2015.
[15] Zoppi T, Ceccarelli A, Bondavalli A. Exploring anomaly detection in systems of systems. Proceedings of the Symposium on Applied Computing 2017:1139–46 April. ACM.

[16] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the kdd cup 99 data set. Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEESymposium on. IEEE 2009:1–6.

[17] Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security 2012;31(3):357–74.

[18] Campos GO, Zimek A, Sander J, Campello RJ, Micenkova B, Schubert E, et al. On the evaluation of outlier detection: measures, datasets, and an empirical study. Lernen, Wissen, Daten, Analysen 2016. CEUR workshop proceedings 2016.

[19] Boughorbel Sabri, Jarray Fethi, El-Anbari Mohammed. "Optimal classifier for imbalanced data using matthews correlation coefficient metric. PLoS ONE 2017;12(6):e0177678.

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

[21] Source Files (online), github.com/tommyippoz/Miscellaneous-Files/blob/ master/SupplementaryMaterial_JISA.zip, accessed: 2019-09-20

[22] Elki data mining. elki-project.github.io 2018-05-30 accessed.

[23] “Weka 3: Data mining software in java”, www.cs.waikato.ac.nz/~ml/weka/, accessed: 2019-07-20

[24] Leung K, Leckie C. Unsupervised anomaly detection in network intrusion detection using clusters. Proc. of the Twenty-eighth Australasian conference on Computer Science 2005;38:333–42 January. Australian Computer Society, Inc.

[25] He S, Zhu J, He P, Lyu MR. Experience report: system log analysis for anomaly detection. In Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on 2016:207–18 October. IEEE.

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

[27] Gruschka N, Jensen M. Attack surfaces: a taxonomy for attacks on cloud services. in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. IEEE 2010:276–9.

[28] J. Syversen, “Method and apparatus for defending against zero-day worm-based attacks,” Apr. 24 2008, US Patent 11/632,669. http://www.google.com/patents/US20080098476 [Online]

[29] Bilge L, Dumitras T. Before we knew it: an empirical study of zero-day attacks in the real world. Proceedings of the 2012 ACM conference on Computer and communications security. ACM 2012:833–44.

[30] Lippmann Richard, Haines Joshua W, Fried David J, Korba Jonathan, Das Kumar. The 1999 darpa offline intrusion detection evaluation. Computer networks 2000;34(4):579–95 2000.

[31] Fontugne Romain, Borgnat Pierre, Abry Patrice, Fukuda Kensuke. "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. Proc. of the 6th International Conference 2010:8 ACM.

[32] Song, J., Takakura, H., & Okabe, Y. (2006). Description of kyoto university benchmark data. Available at link: http://www.takakura.com/Kyoto_data/ BenchmarkData-Description-v5.pdf.

[33] Haider W, Hu J, Slay J, Turnbull BP, Xie Y. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. Journal of Network and Computer Appl 2017;87:185–92 2017.

[34] Rosset S, Inger A. KDD-cup 99: knowledge discovery in a charitable organization’s donor database. SIGKDD Explorations 2000;1(2):85–90.

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

[36] Zoppi T, Ceccarelli A, Bondavalli A. “Evaluation of anomaly detection algorithms made easy with reload. Proceedings of the 30th Int. Symposium on Software Reliability Engineering (ISSRE 2019:446–55 October. IEEE. doi:10.1109/ISSRE.2019.00051.

[37] Rodriguez Juan D, Perez Aritz, Lozano Jose A. "Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE Trans Pattern Anal Mach Intell 2010;32(3):569–75.

[38] Saeys Y, Abeel T, Van de Peer Y. Robust feature selection using ensemble feature selection techniques. Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2008:313–25 September.

[39] Bonett Douglas G. Confidence interval for a coefficient of quartile variation. Comput Stat Data Anal 2006;50(11):2953–7.

[40] Cisco Security Advisory, Cisco nx-os malformed arp header denial of service vulnerability, [Online]. Available at https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/Cisco-SA-20150901-CVE-2015-6277, accessed: 2019-07-20

[41] Yu W, Wang X, Calyam P, Xuan D, Zhao W. “Modeling and detection of camouflaging worm. Transactions on Dependable and Secure Computing 2010;8(3):377–90 IEEE.

[42] Gadge J, Patil AA. Port scan detection. in 2008 16th ieee international conference on networks. IEEE 2008:1–6 December.

[43] Curry E, Hasan S, Pavlopoulou N, Zaarour T, et al. Grand challenge: automatic anomaly detection over sliding windows. Proc. of the 11th ACM International Conference on Distributed and Event-based Systems. ACM 2017.

[44] Zoppi T, Ceccarelli A, Bondavalli A. An initial investigation on sliding windows for anomaly-based intrusion detection. to appear at IEEE SERVICES Workshop on Cyber Security & Resilience in the Internet of Things (CSRIoT) 2019 July.

[45] Zoppi Tommaso, Ceccarelli Andrea, Bondavalli Andrea. "MADneSs: a multi-layer anomaly detection framework for complex dynamic systems. IEEE Trans Dependable Secure Comput 2019. doi:10.1109/TDSC.2019.2908366.

[46] Nostro Nicola, Bondavalli Andrea, Silva Nuno. Adding security concerns to safety critical certification. in software reliability engineering workshops (ISSREW). 2014 IEEE Int. Symposium on. IEEE 2014:521–6.

[47] OWASP. 2018. Open web application security project. www.owasp.org/index.php/Main_Page. Accessed: 2019-07-20

[48] Falcão F, Zoppi T, Silva CBV, Santos A, Fonseca B, Ceccarelli A, Bondavalli A. Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing 2019:318–27 April. ACM.

[49] Chicco Davide. "Ten quick tips for machine learning in computational biology. BioData Min 2017;10(1):35.

[50] McKinney Wes. Python for data analysis: data wrangling with pandas, numpy, and IPython. O’Reilly Media, Inc 2012.

[51] Ding Xuemei, Li Yuhua, Belatreche Ammar, Maguire Liam P. An experimental evaluation of novelty detection methods. Neurocomputing 2014;135:313–27 2014.

[52] Ingham Kenneth L, Inoue Hajime. Comparing anomaly detection techniques for http. In International Workshop on Recent Advances in Intrusion Detection. 2007:42–62.

[53] Gharib M, Bondavalli A. On the evaluation measures for machine learning algorithms for safety-critical systems. In the 15th European Dependable Computing Conference (EDCC 2019 IEEE, To Appear.

[54] Mori M, Ceccarelli A, Zoppi T, Bondavalli A. On the impact of emergent properties on sos security. In 2016 11th System of Systems Engineering Conference (SoSE) 2016:1–6 June. IEEE.

[55] Abubakar Adamu I, et al. A review of the advances in cyber security benchmark datasets for evaluating data-driven based intrusion detection systems. Procedia Comput Sci 2015;62:221–7.

[56] Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P, Bengio S. Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research 2010;11:625–60.

[57] Huang SY, Yu F, Tsaih RH, Huang Y. Resistant learning on the envelope bulk for identifying anomalous patterns. Proceeding of the 2014 International Joint Conference on Neural Networks (IJCNN 2014.

[58] Sheikhan M, Jadidi Z, Farrokhi A. Intrusion detection using reduced–size rnn based on feature grouping. Neural Computing and Applications 2012;21(6):1185–90.

[59] Shoemaker L, Hall LO. Anomaly detection using ensembles. Proceedings of the 10th Multiple Classifier Systems International Workshop (MCS 2011:6–15.

[60] Tenenboim-Chekina L, Rokach L, Shapira B. Ensemble of feature chains for anomaly detection. Proceedings of the 11th Multiple Classifier Systems International Workshop (MCS 2013:295–306.

[61] Tsaih Rua-Huan, Huang Shin-Ying, Lian Mao-Ci, Yennun Huang. “ANN mechanism for network traffic anomaly detection in the concept drifting environment. IEEE DSC 2018 2018:1–6 2018.