Received May 24, 2021, accepted June 16, 2021, date of publication June 21, 2021, date of current version July 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3090957

Unsupervised Algorithms to Detect Zero-Day Attacks: Strategy and Application

TOMMASO ZOPPI, ANDREA CECCARELLI, AND ANDREA BONDAVALLI, (Member, IEEE)
Department of Mathematics and Informatics, University of Florence, 50134 Florence, Italy

Corresponding author: Tommaso Zoppi (tommaso.zoppi@unifi.it)

This work was supported in part by the H2020 Programme through the Marie Skłodowska-Curie (ADVANCE) Project under Grant 823788, and in part by the Regione Toscana SPaCe Project under Grant PORFESR2014-2020.

ABSTRACT In the last decade, researchers, practitioners and companies struggled for devising mechanisms to detect cyber-security threats. Among others, those efforts originated rule-based, signature-based or supervised Machine Learning (ML) algorithms that were proven effective for detecting those intrusions that have already been encountered and characterized. Instead, new unknown threats, often referred to as zero-day attacks or zero-days, likely go undetected as they are often misclassified by those techniques. In recent years, unsupervised anomaly detection algorithms showed potential to detect zero-days. However, dedicated support for quantitative analyses of unsupervised anomaly detection algorithms is still scarce and often does not promote meta-learning, which has potential to improve classification performance. To such extent, this paper introduces the problem of zero-days and reviews unsupervised algorithms for their detection. Then, the paper applies a question-answer approach to identify typical issues in conducting quantitative analyses for zero-days detection, and shows how to setup and exercise unsupervised algorithms with appropriate tooling. Using a very recent attack dataset, we debate on i) the impact of features on the detection performance of unsupervised algorithms, ii) the relevant metrics to evaluate intrusion detectors, iii) means to compare multiple unsupervised algorithms, iv) the application of meta-learning to reduce misclassifications. Ultimately, v) we measure detection performance of unsupervised anomaly detection algorithms with respect to zero-days. Overall, the paper exemplifies how to practically orchestrate and apply an appropriate methodology, process and tool, providing even non-experts with means to select appropriate strategies to deal with zero-days.

INDEX TERMS Zero-day attacks, intrusion detection, machine learning, anomaly detection, RELOAD, security, unsupervised learning, cyber-attacks.

I. INTRODUCTION

It is undeniable that new cyber-attacks are being continuously crafted against essentially any kind of system and service [29], [30], [32], [33]. Attacks have different characteristics: they may exploit known vulnerabilities, overload a system, deliver or install malicious software, etc. Throughout years, rule-based, signature-based and supervised Machine Learning (ML) algorithms [2], [11], [54] have proven to be effective to detect known and fully characterized attacks that show distinguishable patterns, often referred to as signatures or fingerprints. Amongst many other things, it has been shown that attacks may alter memory [31], bytes exchanged through the network [5], packets routing [39], communication buses [6], interactions between components [32], system calls [47], active threads and opened files [29]. In the security domain, supervised ML algorithms are commonly adopted to defend against known threats, and they are embedded into antiviruses or Intrusion Detection Systems (IDSs) which aim to detect attackers that exploit known security breaches [28] or vulnerabilities [30], [31].

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

A. DETECTION OF ZERO-DAY ATTACKS

However, many IDSs show deficiencies in identifying novel, zero-day attacks [33]. These attacks exploit either new vulnerabilities or known vulnerabilities in novel and different ways and cannot be matched against known signatures. Complexity and dynamicity of systems are rapidly increasing, to the extent that a correct and complete characterization of all possible security threats becomes almost impossible. Moreover, the growth of hacking activities often exposes systems to brand new attacks. In such scenario, the likelihood of being targeted by zero-day attacks is getting higher and higher. This motivates a relevant research effort towards detection mechanisms able to efficiently deal with zero-day attacks, such as anomaly detectors [1]–[4], [40].

Differently from signature-based approaches, which require detailed knowledge on each attack, unsupervised anomaly detection algorithms first model the normal (expected) behavior of a system. Then, they use this knowledge to find patterns in data that do not conform to the model of such expected behavior: these patterns are called anomalies [1]. Unsupervised algorithms infer patterns from a training set and discover the underlying structure of the data without reference to known outcomes (i.e., labels are unknown at training time). Instead, they assume that ongoing attacks temporarily alter the values of system indicators with respect to their expected values. This way, they learn a model that is decoupled from labels assigned to data points in the training set and therefore fit the detection of zero-day attacks [4], [37], [40].

B. MOTIVATION

Unfortunately, it is acknowledged that unsupervised anomaly detection algorithms may show poor detection performance [78]–[80] when used as the sole or main instrument for intrusion detection. In particular, they are likely to generate a high amount of False Positives (the detector raises a security alert but no attacks are happening) and False Negatives (attacks going undetected), thus lowering correct classifications as True Positives or True Negatives. On the other hand, they have shown an discussed superiority in detecting zero-days, therefore a sensible strategy appears to create a synergy between supervised ML algorithms and unsupervised ones to build effective IDSs that deal with both known and zero-day attacks [83]–[85].

Regardless of the many different ways to combine the two approaches, employing an unsupervised anomaly detection algorithm without properly optimize its behavior will lead to poor detection performance and drive the encompassing IDS towards misclassifications, with obvious detrimental effects. Consequently, researchers and practitioners have to perform quantitative analyses to select the most appropriate unsupervised algorithm for a given system or use case. Unfortunately, frameworks or supporting tools that perform quantitative analyses mostly implement supervised ML algorithms, whereas unsupervised algorithms are scattered in different libraries. Additionally, the support to meta-learning [42], [43], which was recently proven effective in reducing misclassifications of unsupervised algorithms [41], [46], is actually scarce. Those aspects complicate the execution of different algorithms with the same methodology, because different tools have their own way to perform optimizations, normalizations, batching or to derive additional features.

To such extent, this paper highlights the problem of zero-day attacks and explains why unsupervised algorithms can detect them. Then, it reviews the most common research questions that originate when comparing unsupervised algorithms for intrusion detection, including the usage of meta-learning. Afterwards, the paper analyses a recent, public attack dataset using an open source tool tailored for unsupervised anomaly detection. This shows how relevant research questions can be easily answered, originating unsupervised algorithms with excellent detection performance and also encouraging personnel with limited experience to correctly approach and deal with zero-day attacks.

C. PAPER STRUCTURE

This paper is structured as follows: Section II overviews the problem of zero-day attacks, while Section III presents anomaly detection algorithms for unsupervised intrusion detection. Section IV describes best practices and introduces some research questions that drive quantitative analyses of unsupervised algorithms. Section V reports on tooling tailored to perform such analyses. Section VI describes the quantitative comparison of unsupervised anomaly detection algorithms applied to a recent dataset for network intrusion detection. This shows how the methodology, the process and tooling can be practically applied, even by non-experts. Finally, Section VII wraps up and concludes the paper.

II. INTRUSION DETECTION AND ZERO-DAYS

A. SECURITY AND CYBER-ATTACKS

The U.S.A. Committee on National Security Systems Glossary defines cybersecurity as *prevention of damage to, protection of, and restoration of computers, electronic communications systems or services, and wire communication, including information contained therein, to ensure availability, integrity, authentication, confidentiality, and nonrepudiation* [72].

Regardless of their characteristics, attacks should be timely identified to block an ongoing attack or protect critical assets. Different attacks may be crafted against different kinds of systems [10], [29], [32]. Agencies such as ENISA [48] highlight attacks that use web services, whose main vectors are Browser Exploits, Drive-by-Download, malicious URLs or SQL-injection (SQLi). Other attacks are frequently created and delivered through spam or phishing attacks: over 90% of malware infections in organizations originate from phishing attacks [47]. As a last example, attacks to system availability aim at denial of service [81], or they exploit vulnerabilities of the UDP, TCP and ICMP network protocols, possibly through the orchestration of botnets.

These attacks are generally detected by means of rule-based, signature-based or supervised machine learning techniques implemented into firewalls, which have become more and more effective in detecting these known attacks.

B. UNKNOWN OR ZERO-DAY ATTACKS

Unfortunately, most firewalls cannot effectively deal with *zero-day attacks* [33] (also just called zero-days). The reader can associate a zero-day attack (or unknown exploit) to a door of a garage that the owner did not lock correctly. Whenever thieves discover that the door is unlocked, they can step into the garage, steal or damage many things and then close the door again. Such exploit may go unnoticed for many hours or days. At some point, the owner will discover the problem and correctly lock the door: now, the unknown exploit is no more a zero-day as it was ''patched''. Unfortunately, at that time the damage is already done and the thieves are long gone. A zero-day attack does not necessarily exploit a zero-day vulnerability: in fact, many known vulnerabilities are subject to unknown exploits [34].

In the last decade there was a rapid growth of markets [33], [35] where anyone can ''sell'' zero-days they discovered but not yet put into practice or shared publicly. Those may either be exploited by malicious entities or used to patch a system or a software. For example, demand and price of zero-days in online messaging / video-call software is nowadays growing thanks to the increasing need for virtual meetings, with buyers that are willing to pay hundreds of thousands of dollars [36] to get to know such exploits.

More in general, we cannot avoid zero-days: instead, once a zero-day is discovered by companies, agencies or (ethical) hackers, system owners should race towards patching such exploit. This has two main benefits: it prevents similar exploits and minimizes damages due to past or ongoing attacks. As pointed out by [34], the costs associated to the detection and mitigation of zero-days are undeniably higher than the costs of adopting classic countermeasures as antiviruses. Additionally, their effectiveness is usually situational and therefore the detection of unknown attacks represents a hot and active research topic for both academia and industry.

C. DETECTION OF KNOWN AND UNKNOWN ATTACKS

Intrusion detectors that rely on *supervised* ML algorithms require historical system observations for which a label is known. Supervised algorithms learn a model allowing to classify any new observation (a *data point*) as either collected when a system is targeted by a malicious attack, or during normal operations. For example, the literature reports on the successful usage of Random Forests [44], Support Vector Machines [53], [81], Convolutional Deep Neural Networks [38], [54], [70] for the detection of attacks through the analysis of network traffic, assuming that those attacks are known at training time by the supervised ML algorithms.

On the other hand, *unsupervised* anomaly detection algorithms do not assume any knowledge on the attacks. They model the expected (normal) behavior of the system, and classify any deviation from the normal behavior as anomaly [1] (i.e., a suspect activity, possibly an attack): therefore, they do not distinguish between known attacks and zero-day attacks. As a drawback, their detection performance is often not optimal. In particular, in [80] authors show how a regular neural network outperforms an unsupervised algorithm (i.e., Self-Organizing-Maps [17]), while other works show how unsupervised algorithms result in lower sensitivity (i.e., many false positive) [78] or lower detection performance overall [79].

As a result, unsupervised algorithms are meant to synergize with supervised ML algorithms rather than be used to replace them into IDSs. In [83] authors combine an unsupervised (clustering) strategy to derive additional features which are then provided to the supervised ML algorithm. Other researchers [84] aimed to ''detect and isolate malicious flows from the network traffic and further classify them as a specific type of the known malwares, variations of the known malwares or as a completely new malware strain.'' As a last example, in [85] authors use a stacking ensemble with unsupervised base-level learners and a supervised meta-level learner. It should be noted that combining both approaches is not trivial and does not always result in improved capabilities: some misleading algorithms may let their combination leaning towards a misclassification.

Independently on how the unsupervised algorithm will be combined with supervised solutions it is evident that minimizing the misclassifications of unsupervised anomaly detection algorithms is highly desirable and represents a key challenge for researchers and practitioners working in the security domain.

III. UNSUPERVISED ANOMALY DETECTION

A. FAMILIES OF UNSUPERVISED ALGORITHMS

Different unsupervised anomaly detectors have been proposed throughout years and grouped into families [1], [3]–[5], [40]. We describe them with the support of Figure 1. First, Figure 1a depicts various normal data points and four anomalous data points (supposedly, corresponding to attacks). The successive Figure 1b to Figure 1i graphically describe the different families, reviewed below.

Clustering algorithms [18], [26] partition a dataset by grouping data points in the same cluster if they share similar characteristics. Data points that cannot be assigned to any of the existing clusters, or that do not meet specific inclusion criteria, are anomalous. An example of such behavior is shown in Figure 1b, which identifies 3 separate clusters. In the example, the clustering algorithm identifies two true positives (green tick marks) and two false negatives (red crosses). Similarly, density-based algorithms in Figure 1c estimate the density of a region: data points lying in dense regions of the input space are considered normal, while anomalies are expected in sparse areas.

Classification algorithms identify the binary class of a new data point devising proper boundaries, which may either be linear (Figure 1d) [25] or non-linear (Figure 1e) [15].

Statistical algorithms (Figure 1f) assume that anomalous data points occur in low probability regions of a given statistical distribution (e.g., frequency of histograms [12]) derived during training.

An alternative group of unsupervised algorithms is shown in Figure 1g, which depicts how *angle-based* algorithms [13] perform anomaly detection. They define angles of a data

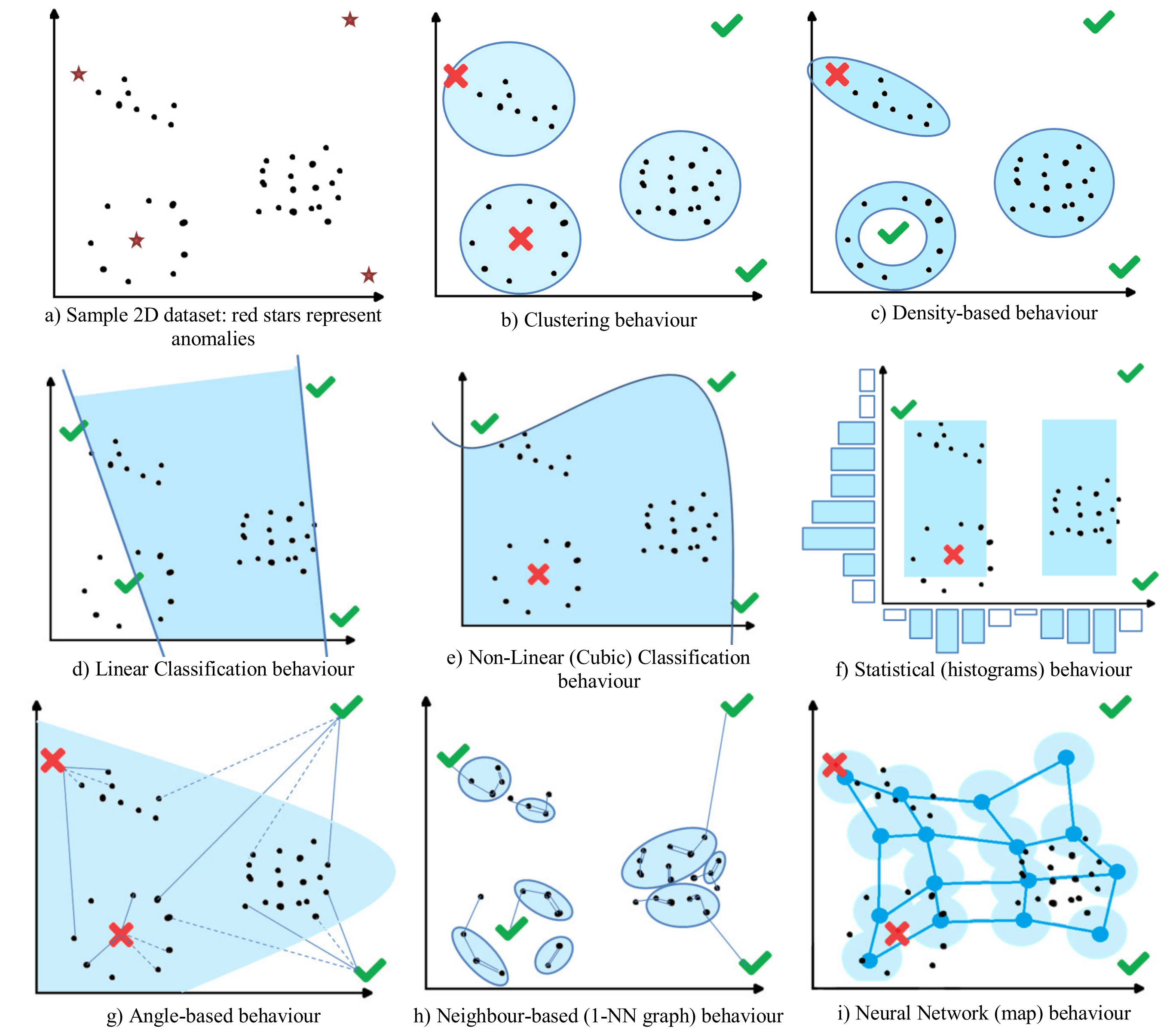

a) Sample 2D dataset: red stars represent anomalies
b) Clustering behaviour
c) Density-based behaviour
d) Linear Classification behaviour
e) Non-Linear (Cubic) Classification behaviour
f) Statistical (histograms) behaviour
g) Angle-based behaviour
h) Neighbour-based (1-NN graph) behaviour
i) Neural Network (map) behaviour

FIGURE 1. **Sample behaviour on a 2-D dataset for algorithms belonging to different families. Light blue areas identify normal regions, crosses indicate false negatives while green ticks indicate true positives. Points lying in the white area that are not marked as ticks identify false positives, while others represent true negatives.**

point with couples of other data points in the training set (see solid and dashed lines in the figure, which define two angles for each data point) and then measure the variance of those angles; anomalies typically result in very small variance.

Instead, *neighbor-based* algorithms [24] classify a data point as anomalous or expected depending on the distance with respect to its nearest neighbor(s). Figure 1h shows how a kNN graph (with k = 1) can be used to perform unsupervised anomaly detection [14] differently from the traditional kNN approaches, which are supervised.

Last, unsupervised *neural networks* [17] produce a two-dimensional, discretized representation (a 4 × 4 map in Figure 1i) of the input space, which during training adapts its weights to map normal data and - consequently - anomalies.

It is worth noticing that there are some unavoidable semantic overlaps among families; for example, neighbors identification is employed to reduce noise and computational complexity in the stochastic ISOS [21], the angle-based FastABOD [13] and in the density-based LOF [23] and COF [20]. DBSCAN [22] and LDCOF [19] build a density-based anomaly detector on top of an internal clustering procedure.

B. UNSUPERVISED INTRUSION DETECTION

Algorithms belonging to different families usually do not exhibit similar detection performance. Although most of those algorithms have a generic, context-independent formulation, they are often more effective to detect specific attacks on specific systems or applications. For example, Leung and Leckie [37] describes how a clustering algorithm outperforms neighbor-based and classification (i.e., One-Class SVM) alternatives to detect attacks in the KDDCupp99 dataset. Instead, [5] describes how neighbor-based algorithms show potential in detecting point anomalies due to attacks in 4 different datasets, whereas One-Class SVM detects contextual anomalies better than other algorithms considered in the study. Differently, studies as [3], [4], [37] focus on the comparison of different (unsupervised) algorithms for anomaly detection in different systems. They quantitatively evaluate detection performance of unsupervised algorithms and values of hyper-parameters that minimize misclassifications.

Additionally, some algorithms may require too much processing time to be efficiently incorporated into specific systems. In particular, while clustering and statistical algorithms usually show linear training time, others require at least quadratic time complexity. However, all unsupervised algorithms apart from neighbor-based have constant test time, and therefore are able to instantaneously process and classify novel data points, which is important for example for the analysis of data streams.

C. UNSUPERVISED META-LEARNING

Recent studies [41], [46] started investigating applications of meta-learning for unsupervised anomaly detection. Meta-learning is defined in [42] as ''the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes''. A meta-learner is a classifier that uses knowledge acquired during base-learning episodes i.e., meta-data, to improve classification performance. Meta-data is usually composed of dataset features and meta-features, which describe attributes of both dataset and base-learners [42].

More specifically, a base-learning process starts by feeding dataset features into one or more ML algorithms to derive one or more models to be used for classification at a first stage. Results of base learners build meta-data that is provided alongside with other features to the meta-classifier, representing the classification result of the whole meta-learner [43]. For instance, bagging builds several base-learners of the same type, trains them using bootstrap replicas of the training set [68] and then combines their individual results through majority voting. Instead, boosting aims at orchestrating several weak learners to build a strong meta-learner [69]. Each weak learner is trained hierarchically to discriminate more carefully specific complex regions in the input space.

Meta-learners were primarily meant to enhance supervised ML algorithms, originating well-known algorithms as Random Forests [44] or ADABoost [45], which orchestrate ensembles of decision trees to reduce misclassifications. Unsupervised boosting and often bagging meta-learners were recently proven [41] to outperform regular unsupervised algorithms for both network intrusion detection and the analysis of biometric data, and therefore should be considered when planning comparison studies in the security domain through anomaly detection.

IV. QUANTITATIVE ANALYSES AND RESEARCH QUESTIONS

The growing relevance of zero-days forces system owners to adopt ad-hoc countermeasures. However, the many possible alternatives for unsupervised intrusion detection make the selection and comparison process tedious, time-consuming and often difficult to execute without knowledge on the insights of the algorithms.

Section IV-A addresses the most suitable metrics to measure detection performance of unsupervised intrusion detectors. Then, Section IV-B discusses on datasets, features, and basic pre-processing activities. Section IV-C explains how to evaluate detection capabilities of algorithms with respect to zero-days. Finally Section IV-D highlights the research questions that usually arise when conducting comparison studies; such questions will find answers in Section VI.

A. METRICS TO CALCULATE DETECTION PERFORMANCE

The detection performance of anomaly detectors is usually scored by means of a confusion matrix [7]. Correct classifications i.e., True Positives (TPs) and True Negatives (TNs) are desirable, while misclassifications, classified as False Positives (FPs) and False Negatives (FNs), should be avoided as much as possible [7], [9].

Security-critical applications as IDSs should primarily focus on reducing FNs, that is, when attacks are being carried out but no anomalies are detected. However, it is also evident that a very suspicious IDS - which heavily reduces the amount of FNs at the price of increasing FPs - may detect many attacks but also generate (too) many false alarms. Therefore, IDSs should be evaluated by focusing on metrics that account for both FPs and FNs, possibly weighting FNs more than FPs.

Consequently, we propose to measure and rank detection performance of unsupervised intrusion detectors by means of two main metrics. The first one is:

- Fβ-Score [7], which becomes a FN-oriented metric when considering $\beta > 1$. In particular, F2-Score doubles the importance of Recall (which accounts for FNs) over Precision (which is linked to FPs) and accounts also for True Positives (TPs).

$$F\beta - Score = (1+\beta^2)\frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

where $Precision = \frac{TP}{TP+FP}$ $\quad Recall = \frac{TP}{TP+FN}$

However, this metric does not account for TNs and therefore offers a partial view on the confusion matrix. This is not adequate [9] in case of unbalanced datasets, i.e., datasets mostly containing normal data points and only few attacks, which are very common in the security domain. Therefore, an additional metric can be used:

- Matthews Correlation Coefficient (MCC) [8], which aggregates all classes of the confusion matrix and correctly measures detection performance even with unbalanced datasets.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)\,(TP + FN)\,(TN + FP)\,(TN + FN)}}$$

B. DATASETS, FEATURES AND PRE-PROCESSING

Recent surveys [54], [55] describe security-related datasets (mostly gathered from publicly available sources) collected by monitoring a given system. Monitoring activities may be either event-based or time-based; under both cases this influences the semantics of features and consequently the interpretation of their values.

Overall, datasets may contain features that are either of i) textual, ii) numeric categorical, or iii) numeric ordinal type. The type of a feature has great relevance in the classification process. Numeric ordinal features (e.g., number of cache hits, bytes sent through the network) express numeric values and therefore can be used for any type of computation. Textual (e.g., name of system call) and numeric categorical (e.g., network port number) features should be individually examined to understand how they can contribute to the classification process.

Some algorithms compute operations such as distance between two feature values or statistics as averages; however, those operations may not be meaningful when processing categorical features as port numbers. It should be mentioned that some algorithms can effectively process categorical values [57]: however, when running comparative studies we should aim at processing features that can be adequately processed by many algorithms rather than optimizing specific characteristics of single algorithms. As a consequence, categorical features should be dropped or re-shaped [56] to become meaningful for anomaly detection.

As a specific case, we highlight a frequent mistake on the usage of IP Address and Port Number as features for intrusion detection. These features should be used only when building an intrusion detector which is for a specific network topology. In the majority of cases, we cannot assume to know the IP address of the attacker(s), or even the exact port to which they will conduct their attack, and therefore we should disregard the usage of IP/port as features.

C. EVALUATING DETECTION PERFORMANCE IN CASE OF ZERO-DAYS

By definition of zero-days, zero-days attacks are not present in any attack dataset. This raises a major question for our study, which specifically aims at evaluating detection capabilities of unsupervised anomaly detection algorithms when exposed to zero-days.

The ability of anomaly detection algorithms to detect zero days can be studied as follows. Algorithms build their model by relying on a training dataset, which represents the knowledge of the algorithm. A type of attack that does not appear in the training set will be a zero-day whenever it appears in the test set as it was never encountered (is unknown to) by the algorithm. For example, let us consider a dataset containing multiple attacks and that can be split in a training set and a test set. If the test set contains one or more types of attacks that do not appear in the training set, such attacks can be considered zero days for the algorithm (which has been trained observing only the attacks present in the training set).

Therefore, it is possible to mimic the occurrence of zero-days by removing specific attacks from the training set, and providing them only during evaluation in the test set. This reproduces the occurrence of zero-days and allows quantitatively evaluating and comparing the performance of algorithms in detecting zero-days.

D. RESEARCH QUESTIONS FOR QUANTITATIVE COMPARISONS

The vast majority of analyses performed toward the study of unsupervised detection algorithms and IDS usually revolve around the following research questions (RQ):

RQ1. Which are the features that contribute the most to a correct classification of data points? In other words, which indicators of the system should be monitored?

RQ2. Which value of hyper-parameters of a given algorithm should be employed to maximize its detection performance?

RQ3. Given a set of potential unsupervised intrusion detectors, which is the best candidate to be deployed in a specific system?

RQ4. Does meta-learning help in reducing misclassifications of unsupervised anomaly detection algorithms?

RQ5. How to evaluate detection performance of unsupervised intrusion detectors with respect to zero-days?

V. A TOOL FOR UNSUPERVISED ANOMALY DETECTION

The paramount importance of tools for the evaluation and comparison of unsupervised anomaly detection algorithms is intuitive. Actually, it is extremely difficult to perform an extensive experimental campaign without supporting frameworks that automate execution of experiments and data analysis.

A. AVAILABLE TOOLS AND FRAMEWORKS

Frameworks as PyTorch [67], Tensorflow [52], CAFFE [75] or even Apache Spark [76] offer a few implementations of unsupervised algorithms such as auto-encoders and clustering algorithms. Those frameworks are primarily meant to optimize performance and scale well in distributed environments by balancing processing load between CPU and GPU. Similarly, SMILE [74], Scikit-Learn [51], and the Statistical and Machine Learning (SML) Toolbox [77] for MATLAB offer many more unsupervised algorithms but again are primarily meant to optimize performance rather than usability. Overall, those frameworks suit the implementation of specific and high-performing algorithms rather than exploratory

TABLE 1. Comparison of different frameworks and tools that run unsupervised machine learning algorithms.

Framework / Tool	Ref	Open Source	GUI	Coding Language	Unsupervised Algorithms	Families of Algorithms	Unsupervised Meta-Learning
Scikit-Learn	[51]	✔		Python	20	5	✔
PyTorch/Tensorflow	[52], [67]	✔		Python	4	2	✔
RELOAD	[27]	✔	✔	Java	17	7	✔
ELKI	[49]	✔	✔	Java	100+	5	
WEKA	[50]	✔	✔	Java	9	2	
RapidMiner	[19]		✔	Java	10	4	
SMILE	[74]	✔		Scala, Java	15	3	
Spark Apache	[76]	✔		Scala, Java, Python	3	3	
SML Toolbox	[77]			MATLAB	10	3	
LibSVM	[53]	✔		C++, Java	1	1	
CAFFE	[75]	✔		C++	1	1	

and comparative studies which embrace different algorithms. Moreover, they do not provide the user with a GUI but require coding, which may prevent non-domain experts to approach unsupervised algorithms.

On the other hand, Java-based tools as RELOAD [27], *ELKI* [49], *WEKA* [50], and *RapidMiner* [19] come as tools that do not force the user to implement code. In fact, the choice of the Java language goes more towards enhancing usability rather than optimizing performance, which is typically done by using lower-level languages as Python, MATLAB and C/C++.

As shown in Table 1, each framework or tool has its own advantages and limitations. Unfortunately, there is no high-performing tool that offers a complete set of unsupervised algorithms through an user-friendly interface. Additionally, only a few options (i.e., Scikit-Learn, PyTorch, Tensorflow and RELOAD) allow to instantiate ensembles of unsupervised algorithms through meta-learning, which improves classification capabilities of unsupervised algorithms as we already discussed in Section III-C.

B. RAPID EVALUATION OF ANOMALY DETECTORS

Amongst all the possible alternatives, our preference lies in RELOAD [27], a tool to perform *Rapid EvaLuatiOn of Anomaly Detectors* that is explicitly shaped to:

- exercise unsupervised anomaly detection algorithms belonging to many different families, even with the support of meta-learning,
- expose a simple GUI that allows also non-experts to perform experiments in a simplified fashion, and
- be lightweight and portable thanks to its Java-based (Version 8+ compatibility) implementation.

RELOAD is an open-source software [63] released under AGPLv3 license that embeds 17 different unsupervised algorithms, either with custom implementation as HBOS [12], LDCOF [19], SDO [16], SOM [17], G-Means [18] or inherited from existing frameworks, namely COF [20], LOF [23], (Fast)ABOD [13], (i)SOS [21], K-Means [26], DBSCAN [22], kNN [24], ODIN [14] from ELKI [49], One-Class SVM [15] from LibSVM [53], and Isolation Forests (iForest, [25]) from WEKA [50].

In a nutshell, the tool wraps existing implementations of unsupervised algorithms from publicly available frameworks, and adds additional algorithms to provide a comprehensive choice. Moreover, it provides a simple GUI for setup operations, such as load a dataset, select features, choose target algorithms and metrics. More sophisticated settings are allowed but not required, providing transparency that allows also non experts to run unsupervised algorithms with the same methodology. Insights of the tool are reported in [27]; in the rest of this paper we will use RELOAD to support experimental analyses and answer the questions of Section IV.

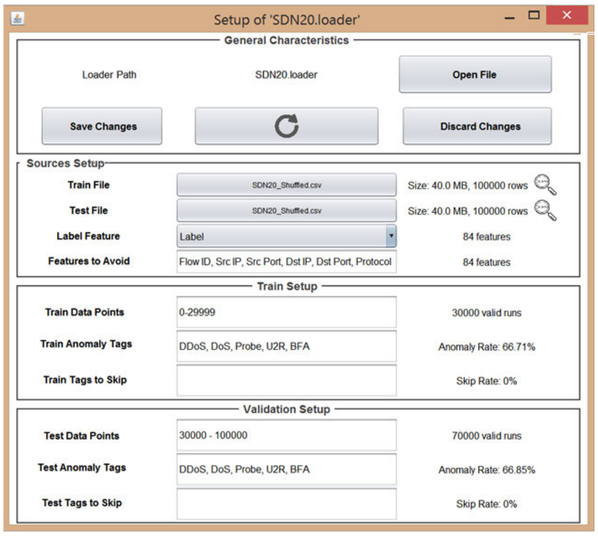
VI. QUANTITATIVE COMPARISON OF UNSUPERVISED ALGORITHMS FOR NETWORK INTRUSION DETECTION

Answers to research questions RQ1 – RQ5 are provided in this section with the support of a publicly available dataset. Out of the many alternatives [54], [55], we perform our experiments using the recent SDN20 [58] dataset, which was released in 2020.

A. DATASET AND PREPROCESSING

SDN20 [58] is a recent dataset built by monitoring a Software Defined Network installed at the University College Dublin (Ireland). Five types of attacks appear in SDN: i) Probe i.e., network scanning, ii) Denial of Service (DoS), iii) Distributed DoS, iv) Brute-Force (BFA) to bypass the username-password login, and v) Exploits (privilege escalation known as U2R). The dataset contains 85 features: 5 textual, 3 categorical and 77 ordinal. Features include classic network data plus Bidirectional Flows where the first packet determines the forward (source to destination) and backward (destination to source) directions.

Starting from the archive made available in [73], we built a portion of this dataset by merging and shuffling the "*Normal-Data.csv*" and "*metasploitable-2.csv*" files contained in archive available at [73]. This allowed creating an unique file of 10^5 data points that includes normal traffic and the



General Characteristics

Loader Path SDN20.loader

Sources Setup

Train File	SDN20_Shuffled.csv	Size: 40.0 MB, 100000 rows
Test File	SDN20_Shuffled.csv	Size: 40.0 MB, 100000 rows
Label Feature	Label	84 features
Features to Avoid	Flow ID, Src IP, Src Port, Dst IP, Dst Port, Protocol	84 features

Train Setup

Train Data Points	0-29999	30000 valid runs
Train Anomaly Tags	DDoS, DoS, Probe, U2R, BFA	Anomaly Rate: 66.71%
Train Tags to Skip		Skip Rate: 0%

Validation Setup

Test Data Points	30000 - 100000	70000 valid runs
Test Anomaly Tags	DDoS, DoS, Probe, U2R, BFA	Anomaly Rate: 66.85%
Test Tags to Skip		Skip Rate: 0%

FIGURE 2. ***SDN20_full*** **loader that connects RELOAD to the SDN20 dataset, with a 30-70 train-test split.**

5 attack types. Such file contains 33.2% of normal data points plus data points related to each of the 5 attacks DDoS (35.9%), DoS (0.5%), Probe (30.2%), U2R (0.01%), BFA (0.1%). We then use RELOAD to setup the *SDN20_full* loader through a dedicated GUI which is also shown in Figure 2. The reader should notice how categorical features *Flow ID, Src IP, Src Port, Dst IP, Dst Port, Protocol* are not included in this analysis (see "Features to Avoid" in the upper portion of Figure 2) in accordance to the discussion in Section IV.B. In addition, we use a 30-70 train-test split.

B. RQ1–IMPACT OF FEATURES

Estimating the impact of features on the detection performance of algorithms is a typical question. Briefly, it is important to understand which features contribute the most to distinguish between normal and anomalous (intrusions) behaviour, leaving out noise or redundant information. A quantitative evaluation of the impact of features can be conducted by means of different filter or wrapper-based feature selection and features ranking strategies [60], [62], which i) do not depend on the ML algorithm to be used at a later stage, and ii) measure similarity of feature values with respect to the label of data points. Algorithm-specific feature selection (e.g., [66] for kNN) or representation learning [65] are usually disregarded for comparison studies as they introduce specific optimizations that complicate discussion.

RELOAD allows selecting features through a dedicated window (see Figure 3, item ①) which shows the available features, their distribution and calculates rankings. Feature ranking results are shown in Figure 3, item ②, and enlarged for readability convenience in Figure 3, item ③. This detailed view shows the features that have the highest Information Gain [59] score (the highest, the better): *Bwd Header Len*, which describes the length of the header of packets in backward direction, scores 0.802. It is worth noticing that different feature selection strategies generate different feature rankings. For example, out of the features in Figure 3, the highest absolute score of Pearson Correlation [61] is achieved by *Pkt Size Avg* (a feature reporting on the average bytes contained in a packet). Depending on the specific needs, different feature ranking strategies may be adopted to drive the selection of the most relevant features and identify extremely noisy features. Overall, the literature acknowledges [59], [60] Information Gain as a feature ranking and selection strategy that suits most of the domains and systems.

C. RQ2–CHOICE OF HYPER-PARAMETERS

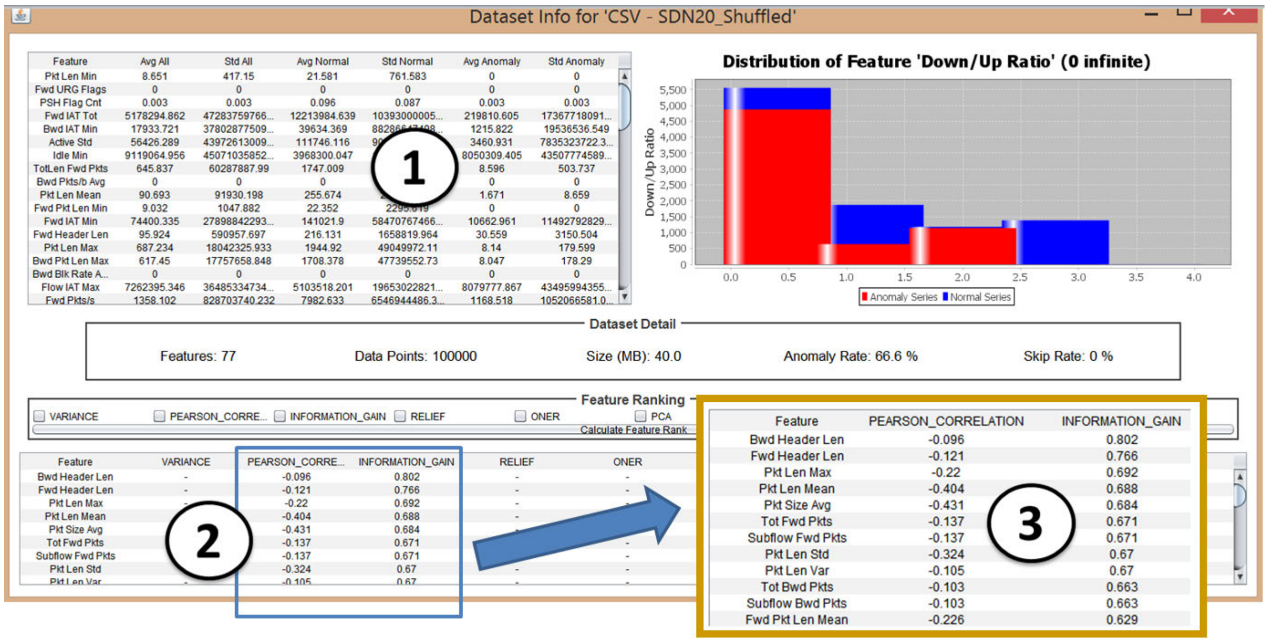
The vast majority of ML algorithms (both supervised and unsupervised), depends on several hyper-parameters to tune the behaviour of the algorithm itself e.g., the size of the neighbourhood for kNN-based algorithms [24] or number of clusters in K-Means [18]. Therefore, when exercising an algorithm we usually aim at devising the value of hyper-parameters that maximize a given classification metric. Finding this exact value (i.e., learning-to-learn [70]) is often not possible, or very time-consuming: therefore, in most of the cases hyper-parameters are chosen out of approximations. For instance, all tools in Table 1 offer the opportunity to run grid or random searches, which consist of building a pool of n potential sets of hyper-parameters that are used to build n instances of the algorithm. If sets of parameters are user-defined, we talk about grid searches; if those sets are randomly generated, we end up conducting random searches. In any case, the values of hyper-parameters that maximise a given metric are chosen as the preferred setup for the algorithm to be used during testing.

Figure 4 shows the GUI of RELOAD that describes detection performance of One-Class SVM on the SDN20 dataset. In addition to the metric scores which are shown in the bottom of the figure, the blue dashed box in the middle of the figure highlights the values of hyper-parameters that maximized MCC (the target metric) at the end of grid searches. Particularly, studies as [15] show how the value *nu* and the type of *kernel* have a major impact on the way the SVM builds the model. Therefore, after setting 4 type of *kernel* = {linear, quadratic, cubic, radial basis function} and 3 values of $nu = \{0.02, 0.1, 0.2\}$, RELOAD conducted a grid search by iterating training 12 times with different kernel and nu values. As shown in the figure, a cubic kernel and $nu = 0.1$ produced the highest MCC value of 0.873 out of the 12 combinations.

D. RQ3–COMPARISON OF ALGORITHMS

The process we followed in RQ2 for One-Class SVM can be repeated to compare the performance of an arbitrary number of algorithms. This process starts by defining one or more metrics that will be used to rank detection performance of algorithms. As explained in Section IV-A, metrics as MCC and F2-Score properly describe detection performance of anomaly detectors and therefore should be the first considered when comparing different intrusion detectors.

Results of our analysis embracing all 17 unsupervised anomaly detection algorithms provided by RELOAD on the SDN20 dataset are shown in Table 2, ranked by


Dataset Info for 'CSV - SDN20_Shuffled'
Distribution of Feature 'Down/Up Ratio' (0 infinite)
Down/Up Ratio
Anomaly Series
Normal Series
1
2
3

Feature	Avg All	Std All	Avg Normal	Std Normal	Avg Anomaly	Std Anomaly
Pkt Len Min	8.651	417.15	21.581	761.583	0	0
Fwd URG Flags	0	0	0	0	0	0
PSH Flag Cnt	0.003	0.003	0.096	0.087	0.003	0.003
Fwd IAT Tot	5178294.862	47283759766...	12213984.639	10393000005...	219810.605	17367718091...
Bwd IAT Min	17933.721	37802877509...	39634.369	88286647498...	1215.822	19536536.549
Active Std	56426.289	43972613009...	111746.116	90[illegible]	3460.931	7835323722.3...
Idle Min	9119064.956	45071035852...	3968300.047	[illegible]	8050309.405	43507774589...
TotLen Fwd Pkts	645.837	60287887.99	1747.009	[illegible]	8.596	503.737
Bwd Pkts/b Avg	0	0	0	[illegible]	0	0
Pkt Len Mean	90.693	91930.198	255.674	2[illegible]	1.671	8.659
Fwd Pkt Len Min	9.032	1047.882	22.352	2295.619	0	0
Fwd IAT Min	74400.335	27898842293...	141021.9	58470767466...	10662.961	11492792829...
Fwd Header Len	95.924	590957.697	216.131	1658819.964	30.559	3150.504
Pkt Len Max	687.234	18042325.933	1944.92	49049972.11	8.14	179.599
Bwd Pkt Len Max	617.45	17757658.848	1708.378	47739552.73	8.047	178.29
Bwd Blk Rate A...	0	0	0	0	0	0
Flow IAT Max	7262395.346	36485334734...	5103518.201	19653022821...	8079777.867	43495994355...
Fwd Pkts/s	1358.102	828703740.232	7982.633	6546944486.3...	1168.518	1052066581.0...

Dataset Detail

Features: 77 Data Points: 100000 Size (MB): 40.0 Anomaly Rate: 66.6 % Skip Rate: 0 %

Feature Ranking

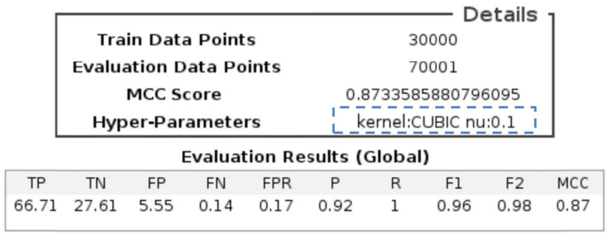
VARIANCE PEARSON_CORRE... INFORMATION_GAIN RELIEF ONER PCA

Calculate Feature Rank

Feature	VARIANCE	PEARSON_CORRE...	INFORMATION_GAIN	RELIEF	ONER
Bwd Header Len	-	-0.096	0.802	-	-
Fwd Header Len	-	-0.121	0.766	-	-
Pkt Len Max	-	-0.22	0.692	-	-
Pkt Len Mean		-0.404	0.688	-	
Pkt Size Avg		-0.431	0.684	-	
Tot Fwd Pkts		-0.137	0.671		-
Subflow Fwd Pkts		-0.137	0.671		-
Pkt Len Std		-0.324	0.67	-	-
Pkt Len Var	-	-0.105	0.67	-	-

Feature	PEARSON_CORRELATION	INFORMATION_GAIN
Bwd Header Len	-0.096	0.802
Fwd Header Len	-0.121	0.766
Pkt Len Max	-0.22	0.692
Pkt Len Mean	-0.404	0.688
Pkt Size Avg	-0.431	0.684
Tot Fwd Pkts	-0.137	0.671
Subflow Fwd Pkts	-0.137	0.671
Pkt Len Std	-0.324	0.67
Pkt Len Var	-0.105	0.67
Tot Bwd Pkts	-0.103	0.663
Subflow Bwd Pkts	-0.103	0.663
Fwd Pkt Len Mean	-0.226	0.629

FIGURE 3. **Visual analytics for datasets and feature ranking through RELOAD. The user can visualize details of the dataset ① and rank features through feature selectors ② and ③.**



Details

Train Data Points	30000
Evaluation Data Points	70001
MCC Score	0.8733585880796095
Hyper-Parameters	kernel:CUBIC nu:0.1

Evaluation Results (Global)

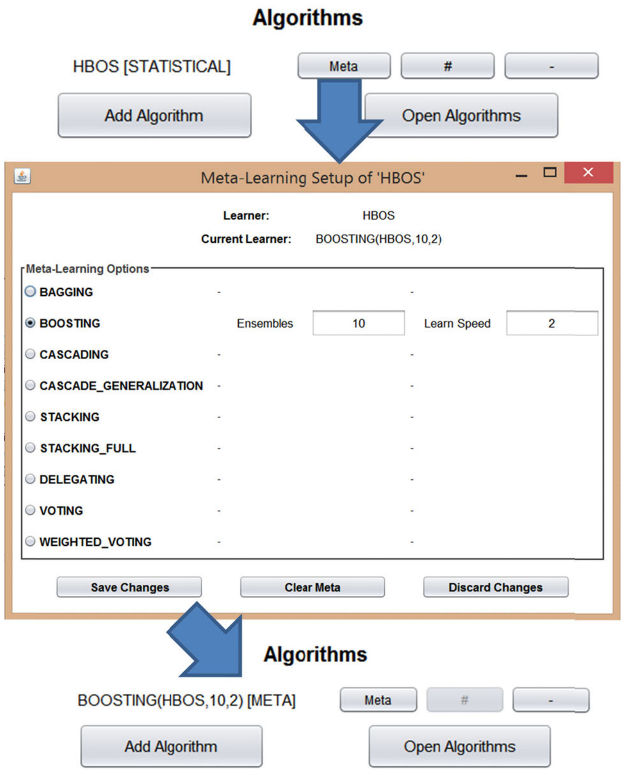
TP	TN	FP	FN	FPR	P	R	F1	F2	MCC
66.71	27.61	5.55	0.14	0.17	0.92	1	0.96	0.98	0.87

FIGURE 4. **Results of one-class SVM on the *SDN20_full* loader with RELOAD. Blue box highlights the result of grid search.**

decreasing MCC. The table reports, for each algorithm, its name and family(ies), the values of hyper-parameters obtained through grid searches, the confusion matrix (TP, TN, FP, FN), Precision, Recall, F2-Score and MCC. Such table, which is provided as output of RELOAD both in the GUI and in a CSV file, allows comparing detection performance of many algorithms according to a common methodology and using the same train set and test set. It is worth noticing that different algorithms exhibit different metric scores, and that algorithms belonging to specific families often show similar behaviour. Classification algorithms as One-Class SVM [15] and iForest [25] show the highest MCC, while statistical algorithms as HBOS [12], SOS and iSOS [21] do not excel. Instead, F2-Score is usually higher than 0.9, and only few neighbour-based algorithms as kNN [24] and COF [20] show many FNs and, consequently, low Recall and F2-Score.

E. RQ4–IMPACT OF META-LEARNING

To check whether improvements are possible, we extend the comparison in RQ3 and include meta-learners, known for having better detection performance (see Section III.C)



Learner: HBOS

Current Learner: BOOSTING(HBOS,10,2)

Meta-Learning Options

○ BAGGING	-		-	
◉ BOOSTING	Ensembles	10	Learn Speed	2
○ CASCADING	-		-	
○ CASCADE_GENERALIZATION	-		-	
○ STACKING	-		-	
○ STACKING_FULL	-		-	
○ DELEGATING	-		-	
○ VOTING	-		-	
○ WEIGHTED_VOTING	-		-	

FIGURE 5. **GUI for instantiating meta-learning in RELOAD.**

than basic unsupervised algorithms. RELOAD allows creating meta-learners as Bagging, Boosting, Stacking, Cascading, Voting, Arbitrating, and Delegating through the dedicated GUI which is shown in Figure 5. According to [41], we instantiate Bagging and Boosting ensembles of 10 items each to be exercised and compared with

TABLE 2. Detection performance of different unsupervised anomaly detection algorithms on the *SDN20_full* loader.

Algorithm	Ref	Family	Hyper-Parameters	TP	TN	FP	FN	Precision	Recall	F2-Score	MCC
iForest	[25]	Classification	samples:500, trees:5	65.717	29.407	3.746	1.130	0.946	0.983	0.975	0.889
One-Class SVM	[15]	Classification	kernel:CUBIC, nu:0.1	66.710	27.605	5.547	0.137	0.923	0.998	0.982	0.873
SDO	[16]	Density	q:0.1 x:5 k:50	65.343	25.547	7.606	1.504	0.896	0.977	0.960	0.793
G-Means	[18]	Clustering	*no params*	66.848	23.515	9.637	0.000	0.874	1.000	0.972	0.787
K-Means	[26]	Clustering	k:10	66.848	23.495	9.657	0.000	0.874	1.000	0.972	0.787
LDCOF	[19]	Density, Clustering	k:10, gamma:0.3	66.848	23.294	9.858	0.000	0.871	1.000	0.971	0.783
SOM	[17]	Neural Network	min_a:0.1, dec:0.9, base_a:0.6	66.846	23.195	9.957	0.001	0.870	1.000	0.971	0.780
FastABOD	[13]	Angle, Neighbour	k:50	66.537	22.046	11.107	0.310	0.857	0.995	0.964	0.746
LOF	[23]	Density, Neighbour	k:2000	60.542	26.285	6.867	6.306	0.898	0.906	0.904	0.702
ODIN	[14]	Neighbour	k:2000	60.648	25.718	7.434	6.200	0.891	0.907	0.904	0.690
iSOS	[21]	Statistical, Neighbour	phi:0.1, k:5	66.743	18.984	14.169	0.104	0.825	0.998	0.958	0.684
DBSCAN	[22]	Clustering, Density	eps:1000, minPts:1	63.943	20.519	12.634	2.904	0.835	0.957	0.929	0.640
SOS	[82]	Statistical	h:20	60.056	22.487	10.666	6.791	0.849	0.898	0.888	0.597
COF	[20]	Density, Neighbour	k:50	49.704	29.285	3.867	17.144	0.928	0.744	0.774	0.592
ABOD	[13]	Angle	*no params*	66.499	14.229	18.924	0.349	0.778	0.995	0.942	0.566
kNN	[24]	Neighbour	k:10	35.497	33.152	0.000	31.351	1.000	0.531	0.586	0.522
HBOS	[12]	Statistical	k:50	66.427	10.740	22.413	0.420	0.748	0.994	0.932	0.475

TABLE 3. F2-score and MCC scores of SDO, HBOS, COF (boosting ensemble) and ODIN (bagging ensemble) on 8 different loaders from the same SDN20 dataset. Each loader exposes different types of attacks and zero-days in the test set.

Loader Code	Train Set	Test Set		SDO		HBOS		Boosting COF		Bagging ODIN	
	Attacks	Known Attacks	Zero-Days	F2-Score	MCC	F2-Score	MCC	F2-Score	MCC	F2-Score	MCC
T5_K5_Z0 (SDN20_full)	DoS, DDoS, BFA, Probe, U2R	DoS, DDoS, BFA, Probe, U2R	-	0.960	0.793	0.932	0.475	0.995	0.986	0.958	0.682
T3_K3_Z2	DoS, Probe, U2R	DoS, Probe, U2R	DDoS, BFA	0.928	0.799	0.885	0.520	0.995	0.986	0.964	0.727
T3_K0_Z2	DoS, Probe, U2R	-	DDoS, BFA	0.909	0.756	0.920	0.550	0.990	0.983	0.943	0.687
T3_K0_Z1	DoS, Probe, U2R	-	DDoS	0.973	0.808	0.941	0.576	0.998	0.992	0.955	0.705
T2_K2_Z3	DDoS, BFA	DDoS, BFA	DoS, Probe, U2R	0.956	0.792	0.782	0.532	0.989	0.971	0.958	0.682
T2_K0_Z3	DDoS, BFA	-	DoS, Probe, U2R	0.917	0.733	0.731	0.472	0.977	0.947	0.913	0.699
T2_K0_Z2	DDoS, BFA	-	DoS, Probe	0.918	0.737	0.733	0.472	0.978	0.949	0.915	0.697
T2_K0_Z1	DDoS, BFA	-	Probe	0.918	0.734	0.736	0.477	0.986	0.956	0.918	0.699
			St.Dev	0.025	0.032	0.096	0.041	0.008	0.018	0.022	0.015

unsupervised algorithms. Figure 6 plots scores of MCC and F2-Score into two separate charts for i) the 17 algorithms from RQ3, ii) bagging and iii) boosting ensembles.

A detailed analysis of these scores should be driven by the specific requirements of the study and by the characteristics of the system under investigation. For example, we may comment that Boosting ensembles usually outperform regular unsupervised algorithms: the green line with crosses in Figure 6 is above the blue diamonds in the majority of cases. According to the scores in Figure 6, it turns out evident how boosting ensembles of either ABOD or COF show the highest MCC and F2-Score scores (i.e., green lines for plots in the figure are close to the maximum of 1 for both metrics) and therefore are good candidates to perform intrusion detection in the SDN20 dataset. Noticeably, Boosting(COF) achieves Precision, Recall, F2-Score, and MCC of {0.997, 0.994, 0.995, 0.986}. Those scores are decisively better than those of the best algorithm in Table 2 (i.e., iForest, which achieves 0.946, 0.983, 0.975, 0.889} for the metrics above), and clearly remark how meta-learning has potential to dramatically improve classification performance of unsupervised anomaly detection algorithms and should always be considered in comparison studies.

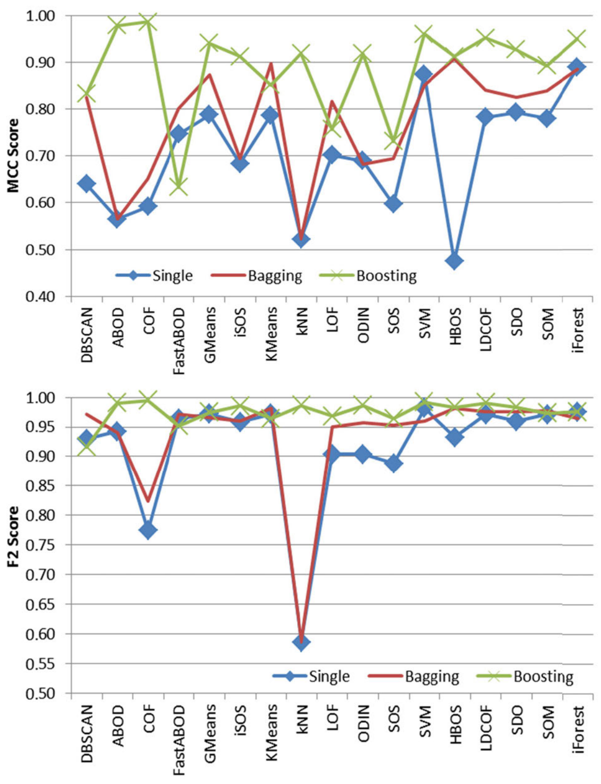


FIGURE 6. MCC (up) and F2-score (down) values for unsupervised algorithms (blue diamonds), ii) bagging ensemble (red line), and iii) boosting ensemble (green crosses).

F. RQ5–DEALING WITH UNKNOWNS

Last, but not least, we show how to evaluate the effectiveness of unsupervised algorithms in detecting zero-days according to the discussion in Section IV-C.

Table 3 reports F2-Score and MCC values obtained by exercising 2 unsupervised algorithms (i.e., HBOS, SDO), a Bagging ensemble of ODIN, and a Boosting ensemble of COF using different loaders (i.e., subsets) of the SDN20 dataset. According to the methodology previously described, each loader exposes different zero-days and is associated to a code *Ta_Kb_Zc* where a) is the number of different types of attacks in the training set, b) is the number of types of attacks that occur both in training and test set (i.e., known attacks), while c) is the number of zero-days, i.e., attacks which appear only in the test set. For example, the loader T3_K3_Z2 builds a training set with normal data and 3 types of attacks (T3), whereas the test set contains 5 types of attacks: 3 known (K3) and 2 zero-days (Z2).

For each loader, Table 3 shows the code, types of attacks that respectively appear in the train and test set, and then reports metric scores of algorithms. The first row reports the loader T5_K5_Z0 in which no zero-days appear in the test set (Z0). Instead, the 2nd, 3rd and 4th row of Table 3 detail three loaders whose training set includes normal data, DoS, Probe and U2R attacks. In these cases, the occurrence (if any) of DDoS or BFA in the test set is a zero-day for the algorithms.

Noticeably, in some cases metric scores even improve when using loaders that expose zero-days (2nd to 8th row) compared to the case with no zero-days (first row of Table 3). In particular, MCC achieved by HBOS using the T3_K3_Z2 loader (0.520, 2nd row of Table 3) is slightly higher than the MCC obtained using T5_K5_Z0 (0.475, first row of the same table). Moreover, loaders as T3_K0_Z2, T3_K0_Z1, T2_K0_Z3, T2_K0_Z2, and T2_K0_Z1 only expose normal data and zero-days in the test set. Even for those loaders, metric scores are not clearly lower with respect to setups with less or even no zero-days.

Additionally, we report the standard deviation of metric scores at the bottom of the table. Those low standard deviation values confirm that metric scores obtained by the same algorithm by using different loaders only slightly fluctuate, further remarking the robustness of unsupervised (meta-learning) algorithms to detect zero-days. Ultimately, Table 3 shows that not all meta-learners improve classification performance with respect to basic unsupervised learning algorithms: the SDO algorithm alone has higher metric scores than the Bagging ensemble of ODIN. On the other hand, the Boosting ensemble of COF shows excellent detection capabilities which hold even when dealing with zero-days.

VII. CONCLUSION

This paper motivated the need to adopt unsupervised anomaly detection algorithms as intrusion detectors to deal with zero-day (unknown) attacks. We first elaborated on zero-day attacks, explaining how they differ from known attacks and why intrusion detectors which rely only on rule-based, signature-based and supervised machine learning algorithms may not reliably identify them. Then, we introduced unsupervised anomaly detection algorithms as well as unsupervised meta-learning approaches that can improve their detection performance.

We then addressed the need for allowing specific unsupervised anomaly detector to be installed in a given system. Proper tuning and selection has to be derived according to a precise strategy and its application through appropriate tooling and experimental campaigns. To achieve this, our study summarizes 5 research questions that impact the vast majority of studies related to the detection of zero-day attacks. Taking advantage of a public attack dataset, we then answer the research questions above. Our study ends up showing that it is possible to derive an unsupervised anomaly detection algorithm built on boosting meta-learning which has much better detection performance than regular unsupervised algorithms and is robust to zero-day attacks. This paper provides the reader with immediate and readily available means to elaborate on unsupervised algorithms and apply them to protect their systems, even against zero-day attacks.

Particularly, we showed how the adoption of meta-learning has the potential to dramatically improve detection performance. This opens an interesting scenario and future works on whether and under which circustances unsupervised meta learning may achieve detection performance that can compete with supervised solutions. To such extent, we foresee a validation process which involves more public datasets in the domain of security, as well as widely used supervised algorithms and deep neural network that suit the analysis of tabular data.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[2] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. IEEE 27th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2016, pp. 207–218.

[3] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS ONE*, vol. 11, no. 4, Apr. 2016, Art. no. e0152173.

[4] T. Zoppi, A. ceccarelli, T. Capecchi, and A. Bondavalli, "Unsupervised anomaly detectors to detect intrusions in the current threat landscape," 2020, *arXiv:2012.11354*. [Online]. Available: http://arxiv.org/abs/2012.11354

[5] T. Zoppi, A. Ceccarelli, L. Salani, and A. Bondavalli, "On the educated selection of unsupervised algorithms via attacks and anomaly classes," *J. Inf. Secur. Appl.*, vol. 52, Jun. 2020, Art. no. 102474.

[6] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[7] 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," in *Proc. CEUR Work-Shop Lernen, Wissen, Daten, Analysen*, Sep. 2016, pp. 1–43.

[8] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PLoS ONE*, vol. 12, no. 6, Jun. 2017, Art. no. e0177678.

[9] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, Dec. 2020.

[10] Y. Chen, Y. Li, X. Q. Cheng, and L. Guo (2006 November), "Survey and taxonomy of feature selection algorithms in intrusion detection system," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Berlin, Germany: Springer, Nov. 2006, pp. 153–167.

[11] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[12] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," in *KI-2012: Poster and Demo Track*, 2012, pp. 59–63.

[13] H.-P. Kriegel, M. S Hubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 444–452.

[14] V. Hautamaki and I. P. K. Franti, "Outlier detection using k-nearest neighbour graph. Pattern Recognition. ICPR 2004," in *Proc. 17th Int. Conf*, vol. 3, Aug. 2004, pp. 430–433.

[15] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proc. ACM SIGKDD Workshop Outlier Detection Description (ODD)*, 2013, pp. 8–15.

[16] F. Iglesias Vazquez, T. Zseby, and A. Zimek, "Outlier detection based on low density models," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 970–979.

[17] T. Kohonen, "Exploration of very large databases by self-organizing maps," in *Proc. Int. Conf. Neural Netw. (ICNN)*, Jun. 1997, pp. PL1–PL6.

[18] G. Hamerly and C. Elkan, "Learning the k in k-means," in *Proc. Neural Inf. Process. Syst.*, 2004, pp. 281–288.

[19] M. Amer and M. Goldstein, "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer," in *Proc. 3rd RapidMiner Community Meeting Conf. (RCOMM)*, 2012, pp. 1–12.

[20] J. Tang, Z. Chen, A. W.-C. Fu, and W. D. Cheung, "Enhancing effctiveness of outlier detections for low density patterns," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springer, 2002, pp. 535–548.

[21] E. Schubert and M. Gertz, "Intrinsic t-stochastic neighbor embedding for visualization and outlier detection," in *Proc. Int. Conf. Similarity Search Appl.*, Cham, Switzerland: Springer, Oct. 2017, pp. 188–203.

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1996, pp. 226–231.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2000, pp. 93–104.

[24] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Reverse nearest neighbors in unsupervised distance-based outlier detection," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1369–1382, May 2015.

[25] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proc. 8th Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.

[26] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.

[27] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Evaluation of anomaly detection algorithms made easy with RELOAD," in *Proc. IEEE 30th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2019, pp. 446–455.

[28] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[29] Y. Zeng, X. Hu, and K. G. Shin, "Detection of botnets using combined host- and network-level information," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2010, pp. 291–300.

[30] B. Fang, Q. Lu, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "EPVF: An enhanced program vulnerability factor methodology for cross-layer resilience analysis," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2016, pp. 168–179.

[31] K. Pan, E. Rakhshani, and P. Palensky, "False data injection attacks on hybrid AC/HVDC interconnected systems with virtual inertia—Vulnerability, impact and detection," *IEEE Access*, vol. 8, pp. 141932–141945, 2020.

[32] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," in *Proc. 43rd Annu. IEEE/IFIP Conf. Dependable Syst. Netw. Workshop (DSN-W)*, Jun. 2013, pp. 1–12.

[33] J. Meakins, "A zero-sum game: The zero-day market in 2018," *J. Cyber Policy*, vol. 4, no. 1, pp. 60–71, Jan. 2019.

[34] *The Cost of Zero-Day Attack Protection*. Accessed: May 23, 2021. [Online]. Available: https://2020infosec.com/the-cost-of-zero-day-attack-protection

[35] *A Zero-Day Guide for 2020: Recent Attacks and Advanced Preventive Techniques*. Accessed: May 23, 2021. [Online]. Available: https://blog.malwarebytes.com/exploits-and-vulnerabilities/2020/06/a-zero-day-guide-for-2020/

[36] *A Zoom Zero-Day Exploit is up for Sale for 500,000*. Accessed: May 23, 2021. [Online]. Available: https://securityboulevard.com/2020/04/a-zoom-zero-day-exploit-is-up-for-sale-for-500000

[37] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proc. 28th Australas. Conf. Comput. Sci.*, vol. 38. Darlinghurst, SYD, Australia: Australian Computer Society, Jan. 2005, pp. 333–342.

[38] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020.

[39] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, and C. Yang, "Traffic engineering in software-defined networking: Measurement and management," *IEEE Access*, vol. 4, pp. 3246–3256, 2016.

[40] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised network intrusion detection systems: Detecting the unknown without knowledge," *Comput. Commun.*, vol. 35, no. 7, pp. 772–783, Apr. 2012.

[41] T. Zoppi, M. Gharib, M. Atif, and A. Bondavalli, "Meta-learning to improve intrusion detection in cyber-physical systems," in *ACM Transactions on Cyber-Physical Systems Special Issue on Artificial Intelligence in CPSs*. New York, NY, USA: Association for Computing Machinery, 2021.

[42] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*. Berlin, Germany: Springer, 2009.

[43] J. Vanschoren, "Understanding machine learning performance with experiment databases," Ph.D. dissertation, Arenberg Doctoral School Sci., Eng. Technol., Katholieke Universiteit Leuven, Brussels, Belgium, 2010.

[44] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[45] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, vol. 96, 1996, pp. 148–156.

[46] J. A. Saez and E. Corchado, "A meta-learning recommendation system for characterizing unsupervised problems: On using quality indices to describe data conformations," *IEEE Access*, vol. 7, pp. 63247–63263, 2019.

[47] (2019). Accessed: May 23, 2021. [Online]. Available: https://enterprise.verizon.com/resources/reports/2019/2019-data-breach-investigations-report-emea.pdf
[48] ENISA. (2020). *Threat Landscape Report*. [Online]. Available: https://www.enisa.europa.eu/news/enisa-news/enisa-threat-landscape-2020
[49] E. Achtert, H. P. Kriegel, and A. Zimek, ''ELKI: A software system for evaluation of subspace clustering algorithms,'' in *Proc. Int. Conf. Sci. Stat. Database Manage.*, Berlin, Germany: Springer, Jul. 2008, pp. 580–585.
[50] S. R. Garner, ''Weka: The waikato environment for knowledge analysis,'' in *Proc. New Zealand Comput. Sci. Res. Students Conf.*, Apr. 1995, pp. 57–64.
[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, ''Scikit-learn: Machine learning in Python,'' *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, and M. Devin, ''Tensorflow: A system for large-scale machine learning,'' in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[53] C.-C. Chang and C.-J. Lin, ''LIBSVM: A library for support vector machines,'' *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
[54] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, ''A taxonomy of network threats and the effect of current datasets on intrusion detection systems,'' 2018, *arXiv:1806.03517*. [Online]. Available: http://arxiv.org/abs/1806.03517
[55] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, ''A survey of network-based intrusion detection data sets,'' *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019.
[56] N. Lee and J.-M. Kim, ''Conversion of categorical variables into numerical variables via Bayesian network classifiers for binary classifications,'' *Comput. Statist. Data Anal.*, vol. 54, no. 5, pp. 1247–1265, May 2010.
[57] J. T. Hancock and T. M. Khoshgoftaar, ''Survey on categorical data for neural networks,'' *J. Big Data*, vol. 7, no. 1, pp. 1–41, Dec. 2020.
[58] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, ''InSDN: A novel SDN intrusion dataset,'' *IEEE Access*, vol. 8, pp. 165263–165284, 2020.
[59] B. Azhagusundari and A. S. Thanamani, ''Feature selection based on information gain,'' *Int. J. Innov. Technol. Exploring Eng.*, vol. 2, no. 2, pp. 18–21, 2013.
[60] A. G. Karegowda, A. S. Manjunath, and M. A. Jayaram, ''Comparative study of attribute selection using gain ratio and correlation based feature selection,'' *Int. J. Inf. Technol. Knowl. Manage.*, vol. 2, no. 2, pp. 271–277, 2010.
[61] H. F. Eid, A. E. Hassanien, T. H. Kim, and S. Banerjee, ''Linear correlation-based feature selection for network intrusion detection model,'' in *Proc. Int. Conf. Secur. Inf. Commun. Netw.*, Berlin, Germany: Springer, Sep. 2013, pp. 240–248.
[62] M. Kuhn and K. Johnson, *Feature Engineering and Selection: A Practical Approach for Predictive Models*. London, U.K.: Chapman & Hall, 2019.
[63] *RELOAD Github*. Accessed: May 23, 2021. [Online]. Available: https://github.com/tommyippoz/RELOAD
[64] S. Džeroski and B. Ženko, ''Is combining classifiers with stacking better than selecting the best one?'' *Mach. Learn.*, vol. 54, no. 3, pp. 255–273, Mar. 2004.
[65] W. Guo, J. Wang, and S. Wang, ''Deep multimodal representation learning: A survey,'' *IEEE Access*, vol. 7, pp. 63373–63394, 2019.
[66] I. M. El-Hasnony, S. I. Barakat, M. Elhoseny, and R. R. Mostafa, ''Improved feature selection model for big data analytics,'' *IEEE Access*, vol. 8, pp. 66989–67004, 2020.
[67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, ''PyTorch: An imperative style, high-performance deep learning library,'' 2019, *arXiv:1912.01703*. [Online]. Available: http://arxiv.org/abs/1912.01703
[68] L. Breiman, ''Bagging predictors,'' *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[69] R. E. Schapire, ''The strength of weak learnability,'' *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, Jun. 1990, doi: 10.1007/BF00116037.
[70] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, ''Learning to learn by gradient descent by gradient descent,'' 2016, *arXiv:1606.04474*. [Online]. Available: http://arxiv.org/abs/1606.04474
[71] B. H. Barlow, ''Unsupervised learning,'' *Neural Comput.*, vol. 1, no. 3, pp. 295–311, Mar. 1989.
[72] *NIST Computer Security Resource Center (Glossary)*. Accessed: May 23, 2021. [Online]. Available: https://csrc.nist.gov/glossary/term/cybersecurity
[73] *Source Files of SDN Dataset*. Accessed: May 23, 2021. [Online]. Available: http://iotseclab.ucd.ie/datasets/SDN/inSDNDataset_CSV.zip
[74] (2014). *SMILE Smile, Haifeng Li*. [Online]. Available: https://haifengl.github.io
[75] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, ''Caffe: Convolutional architecture for fast feature embedding,'' in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678.
[76] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, and J. Freeman, ''Mllib: Machine learning in apache spark,'' *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, 2016.
[77] ''Statistics and machine learning toolbox,'' MathWorks, Natick, MA, USA, Tech. Rep., 2018.
[78] N. S. Arunraj, R. Hable, M. Fernandes, K. Leidl, and M. Heigl, ''Comparison of supervised, semi-supervised and unsupervised learning methods in network intrusion detection system (NIDS) application,'' in *Anwendungen Und Konzepte Der Wirtschaftsinformatik*, 2017.
[79] X. Niu, L. Wang, and X. Yang, ''A comparison study of credit card fraud detection: Supervised versus unsupervised,'' 2019, *arXiv:1904.10604*. [Online]. Available: http://arxiv.org/abs/1904.10604
[80] K. Lee, D. Booth, and P. Alam, ''A comparison of supervised and unsupervised neural networks in predicting bankruptcy of korean firms,'' *Expert Syst. Appl.*, vol. 29, no. 1, pp. 1–16, Jul. 2005.
[81] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, ''An evolutionary SVM model for DDOS attack detection in software defined networks,'' *IEEE Access*, vol. 8, pp. 132502–132513, 2020.
[82] J. H. M. Janssens, F. Huszár, E. O. Postma, and H. J. van den Herik, ''Stochastic outlier selection,'' Tilburg Centre Creative Computing, Herndon, VA, USA, Tech. Rep. 2012-001, 2012.
[83] F. Carcillo, Y.-A. Le Borgne, O. Caelen, Y. Kessaci, F. Oblé, and G. Bontempi, ''Combining unsupervised and supervised learning in credit card fraud detection,'' *Inf. Sci.*, vol. 557, pp. 317–331, May 2021.
[84] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, ''Combining supervised and unsupervised learning for zero-day malware detection,'' in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2022–2030.
[85] T. Zoppi and A. Ceccarelli, ''Prepare for trouble and make it double! supervised—Unsupervised stacking for anomaly-based intrusion detection,'' *J. Netw. Comput. Appl.*, vol. 189, Sep. 2021, Art. no. 103106.

TOMMASO ZOPPI is currently a Research Associate with the University of Florence. He is involved in several European/national funded and even industrial projects. He currently serves as a member of the program committee of several international conferences. His research interests include anomaly detection, security and safety, often applying standards to plan, design, develop, and implement appropriate architectures or software in the domain of critical systems.

ANDREA CECCARELLI received the Ph.D. degree in informatics and automation engineering from the University of Florence, Florence, Italy, in 2012. He is currently a tenured Assistant Professor of computer science at the University of Florence. His research interests include the design, monitoring, and experimental evaluation of dependable and secure systems and systems-of-systems. His scientific activities originated more than 100 articles, which appeared in international conferences, workshops, and journals.

ANDREA BONDAVALLI (Member, IEEE) is currently a Full Professor of computer science at the University of Florence. His research interests include the design and evaluation of resilient and secure systems and infrastructures. His scientific activities originated more than 220 articles appeared in international journals and conferences. He led various national and European projects and has been chairing the program committee in several international conferences. He is a member of the IFIP W. G. 10.4 Working Group on ''Dependable Computing and Fault-Tolerance.''