NAD: Machine Learning Based Component for Unknown Attack Detection in Network Traffic

Mateusz Krzysztoń(✉), Marcin Lew, and Michał Marks

Research and Academic Computer Network (NASK),
Kolska 12, 01-045 Warszawa, Poland
mateuszkr@nask.pl

Abstract. Detection of unknown attacks is challenging due to the lack of exemplary attack vectors. However, previously unknown attacks are a significant danger for systems due to a lack of tools for protecting systems against them, especially in fast-evolving Internet of Things (IoT) technology. The most widely used approach for malicious behaviour of the monitored system is detecting anomalies. The vicious behaviour might result from an attack (both known and unknown) or accidental breakdown. We present a Net Anomaly Detector (NAD) system that uses one-class classification Machine Learning techniques to detect anomalies in the network traffic. The highly modular architecture allows the system to be expanded with adapters for various types of networks. We propose and discuss multiple approaches for increasing detection quality and easing the component deployment in unknown networks by known attacks emulation, exhaustive feature extraction, hyperparameter tuning, detection threshold adaptation and ensemble models strategies. Furthermore, we present both centralized and decentralized deployment schemes and present preliminary results of experiments for the TCP/IP network traffic conducted on the CIC-IDS2017 dataset.

Keywords: Anomaly detection · Machine learning · One-class classification · IoT networks · LoRaWAN security

1 Introduction

According to the current report [1] attacks against corporate networks increases each year. Only in 2021 did the annual number of such attacks rise by 50%. The rapid development of IoT systems is conducive to new unknown attacks. Those unknown attacks are a significant danger for systems due to standard signature-based intrusion detection systems' ineffectiveness for protecting systems against them. Thus, anomaly detection is a commonly used solution to detect malicious behaviour in network traffic [4]. The anomaly in network traffic might result from an attack (both known and unknown) or accidental breakdown.

Distinguishing between normal and abnormal behaviour is difficult due to irregularity of users and external systems behaviour (e.g. different frequency of

J. Kołodziej et al. (Eds.): Cybersecurity of Digital Service Chains, LNCS 13300, pp. 83–102, 2022.
https://doi.org/10.1007/978-3-031-04036-8_4

visiting websites depending on the time of day and marketing campaigns). As a result, detecting anomalies is usually a trade-off between accepting significant irregularities that might result from attack (false-negative errors) and frequent reporting of alarms in situations where the system is working correctly (false-positive errors).

Applying machine learning (ML) to detect anomalies in extensive data set is a well-established approach [18]. At the same time, according to the recent review on anomaly detection [21], cybersecurity is currently one of the most popular fields for applying anomaly detection. When considering anomaly detection, the two types of tasks must be distinguished—outlier detection [10] and novelty detection [23]. In the outlier detection task, the anomaly (i.e. outlier—an object that differs much from the others) is included in the training data. The task objective is to find the most deviant objects in the set. The outlier detection is unsupervised learning due to the lack of labels in the training set. In novelty detection, the training set contains normal (correct, benign) objects only. The task objective is to find anomalies (i.e. novelties—the objects that differ much from those observed before) in the new data. Anomaly detection is considered a semi-supervised learning task as the elements in the training set contain labelled data, but there are no outliers. In the context of ML algorithms applicable for both tasks, the main difference is related to data distribution. In the outlier detection, abnormal objects do not form a dense group (otherwise, those objects are not considered outliers). In contrast, in novelty detection, abnormal objects can form such groups (e.g. samples recorded based on the network traffic during one specific attack).

In this work, we present Net Anomaly Detector (NAD) – the component designed to detect unknown attacks by recognising the network traffic novelties. The NAD module records regular traffic of the system and creates multiple models of normal traffic in the system using ML algorithms. Then module selects the model with the best performance according to the given criteria or combines models with one of the proposed strategies (i.e. ensemble model) and detect anomalies in network traffic using the selected model. The selected base approach is well established and widely used. Thus, we focused on the several features to increase robustness and ease both the deployment of the component in unknown networks and extending the module in the future—known attacks emulation, exhaustive features extraction, hyperparameter tuning, an adaptation of detection threshold and ensemble models strategies. The highly modular architecture allows the system to be expanded with adapters for various types of networks. Additionally, we propose three different component deployment schemes that differ in the level of decentralisation. One of the schemes was used in the actual testbed deployment within GUARD project[1].

[1] GUARD is a project co-funded within Horizon 2020 Funding Programme. The project aims to provide a cybersecurity framework to guarantee reliability and trust for digital service chains. More information can be found on the project website: https://guard-project.eu/.

The chapter is organized as follows. First, we discuss the application of machine learning for novelty detection within the NAD component. Then modular architecture of the NAD component is presented—main modules, variants of component deployment scheme and an exemplary deployment within the GUARD project. Finally, selected results of the experiments conducted on the CIC-IDS2017 dataset are presented.

2 Anomaly Detection with Machine Learning

The main aim of using the NAD component is to define the characteristics of the monitored network traffic, which should allow traffic anomalies detection with reasonably high quality (i.e. both low false-positive (FP) and false-negative (FN) rates). Machine learning is used to create a model of benign network traffic. The result of the learning phase is a model of initial network traffic. The model is used to validate new traffic in real-time—NAD checks whether online traffic fits the created model (i.e. if the traffic does not deviate from benign traffic too much). NAD activates anomaly alert if an outlier in traffic characteristic is detected.

2.1 Machine Learning Methods

The network traffic monitored in a specified location can be defined as a set of consecutive network messages recorded at that point. Timestamp of creation or registering can be assigned to each message. The set of messages can be divided into separate subsets according to some criteria (e.g. into time windows or by address of the sending device). The raw traffic needs to be converted into numeric vectors (a features extraction phase), each one representing a single set of messages. Vectors representing sets of messages recorded during regular (normal, benign, not malformed) system operation are used to create a model of the regular network traffic. The traffic (converted to numeric vectors) that does not match the network model is recognised as an anomaly.

As pointed out, the anomaly detection considered in this work is of the novelty detection type. Thus to train the model, only benign traffic is required. However, in our approach, abnormal traffic samples are also needed to validate models. If abnormal traffic is not available or cannot be produced in the monitored network, the NAD emulation module can be used to generate such traffic.

Training model with samples of one class only is known as One-Class Classification (OOC) [20]. The following ML-based classifiers have been implemented in NAD for solving OOC problems:

- One-Class SVM [30]—version of Support Vector Machine method [13], which builds a hyperplane that separates all or most input data points (representing regular traffic) from the origin (instead of the second class).
- Autoencoder [12]—the type of artificial neural network that can encode input vector and then decode the encoded input vector to get the original value. The autoencoder is built on data points representing regular traffic only.

Hence, only such data points can be encoded and decoded correctly. Other data points representing abnormal traffic after encoding and decoding are not similar to the original values.
- Variational Autoencoder [6]—the type of autoencoder which, instead of constructing latent space explicitly, first learns how to generate distribution depending on input sample (encoder part). The distribution is used to sample latent variable that is then used to reconstruct the input sample (decoder part).
- Local Outlier Factor [11]—to detect anomaly density of training samples around the evaluated sample is calculated and compared with the density of the given number of neighbours. If the density of the evaluated sample is significantly lower than the one of neighbours, then the sample is considered an anomaly.
- Clustering [5]—the input data are divided into one or more clusters (e.g. if network traffic varies at a different time of day). Each cluster is described with mean and radius. If a new data point does not belong to any cluster, it is identified as an anomaly. In most cases, the number of clusters that input data should be divided is unknown. Hence, using clustering methods that do not require an expected number of clusters as an input (e.g. model-based clustering based on finite Gaussian mixture modelling) or simple clustering (e.g. k-means) combined with the elbow method for the best number of cluster detection should be used.

The features selection in OCC problems is challenging due to lack of malicious samples in the training set—it is hard to predict which features will infer the presence of an anomaly. The optimal features set for detecting an attack may depend strongly on the attack type. Hence, the feature selection phase can be omitted in the case of machine learning methods that cope well with high dimensional data. However, this is not the case for all methods, e.g. One-Class SVM. Hence, the following methods were implemented in the NAD for extraction:

- Principal component analysis (PCA) [22]—a dimensionality reduction method, which performs orthogonal linear transformation to transform a set of features to a set of new features. The objective of the change is to keep the maximum possible variance of data while reducing the cardinality of the features set.
- Simple measures selection—a bunch of simple statistical measures are used to rank all features, and the defined number of best features is selected. The following measures were considered: kurtosis, Laplacian score [16], variance, Spectral score [19], the mean distance between points and centroid, inter-cluster distance, inter-quartile range. Following rank aggregation methods for combining rankings acquired by each measure were examined: average, Borda [29] and Dowdall [24].
- Static selection—features are filtered based on a predefined list of feature names to include or exclude.

– Pipeline—the features selection methods proposed above can be combined to create a pipeline, i.e. features set is reduced by each method, and the result is forwarded as the input to the consecutive method.

It could be argued that since malformed traffic is available during the training phase, the problem of attack detection could be considered as the outlier detection problem. However, the malicious traffic samples potentially form dense areas (e.g. traffic originating from one specific type of attack). Additionally, in practice, the traffic recorded in the network that is not under attack may contain contamination forming sparse areas. Such elements may result from, i.a. untypical user behaviour (but not malicious). We would call them *semi-outliers* since they are outliers from the perspective of the machine learning methods, but on the other hand, they are benign from the perspective of network security. However, in the case of binary classification, due to the infrequency and irregularity of those samples, they could be marked as anomalies with a greater probability than emulated malformed traffic injected into the training set.

Semi-outliers cause problems in novelty detection as well. ML algorithms tend to mark those samples as anomalies. Thus, reducing the high false-positive errors rate is one of the biggest challenges in anomaly detection in network traffic. The trade-off between false positive and false negative rates can be controlled with the decision threshold parameter.

2.2 Attack Emulation

Lack of malicious data in the training dataset causes problems with models evaluation and comparison, as detection quality for malicious samples cannot be evaluated. Thus, to overcome those problems, we used simulation techniques to implement attacks emulation. The attack emulation module is responsible for the artificial malformation of the network traffic. The result of the emulation is abnormal traffic that is further used to:

– Tune models hyperparameters,
– Adjust the detection threshold value,
– Compare models.

It needs to be highlighted that applying emulated malicious traffic has drawbacks as well—if the unknown attack is not similar to the ones emulated, it still can be missed by the anomaly detector. Thus emulation attacks should be diverse and representative.

2.3 Adjusting Detection Threshold

The OCC methods create a model of benign traffic. This model is then used to evaluate how well the newly monitored traffic fits the model. The evaluation result can be binary (does or does not fit) or numeric (i.e. score value). The score interpretation differs depending on algorithm type, but generally, it can be interpreted as the indicator of how much the evaluated sample differs from

the normal sample. Binary decision mode is the default behaviour and is based on the threshold. The OCC method calculates the threshold value based on the training data. However, in the case of novelty detection, the OCC method has to guess what threshold will be optimal for detecting anomalies caused by future cyberattacks. In practice, the OCC method calculates some statistical measures for training data to decide how much of that data is abnormal. Such an approach is justified as training data may contain anomalies originating from other sources (more or less similar to the anomalies created by attacks). However, data with anomalies originating from some cyberattacks (real or emulated) within the NAD system is available. Even though such anomalies are mostly not identical to those caused by unknown attacks, still they are potentially more similar than anomalies caused by other sources. Thus those data can be used to support the decision-making process. We decided to implement a mechanism that makes it possible to use malicious data to adjust the classifier threshold value instead of the default one. The benign and malicious traffic samples are scored using the model in the adjusting threshold phase. For calculation of the optimal threshold value of Youden's J statistic [9] or f_β-score is used.

2.4 Ensemble Learning

When malicious traffic is available, the comparison of various algorithms and parameters is possible. However, the quality of unknown attacks detection cannot be predicted with high certainty. Moreover, models that proved to be of lower quality during the validation phase may outperform better models when new attacks occur. Therefore, we implemented in the NAD component several ensemble techniques that increase the stability of detection quality for various attacks. Ensemble learning is a technique to combine multiple models in one compound detector [14]. Thus, in NAD, the outcome of building an anomaly detector can be a bunch of base detectors—one detector for each user-defined configuration space. With each base detector, the weight value is assigned. The weight value is equal to the quality of the detector on the validation set, normalised for all detectors. The creation of an ensemble detector can increase the robustness of the detection. Four basic ensemble strategies were implemented in NAD:

- Non-weighted voting (SV)—each base detector votes if the sample represents an anomaly. The resulting score of the ensemble detector is the number of positive votes divided by a number of base detectors.
- Weighted voting (WV)—each base detector votes if the sample represents an anomaly. The resulting score of the ensemble detector is the sum of weights of base detectors that voted for the anomaly existence.
- Non-weighted scoring (AS)—each base detector scores the sample. The resulting score of the ensemble detector is the average score of all scores.
- Weighted scoring (WS)—each base detector scores the sample. The resulting score of the ensemble detector is a weighted average of all scores.

The important aspect of the two last strategies was to ensure that each base detector uses the same range of scores. Additionally, if two detectors assess the possibility that the sample is an anomaly on a similar level, they should score the sample with a similar value. Each base detector scores the samples with a value from 0 to 1. The scaling factor is learned on the training set. The detection threshold value is adjusted for each ensemble detector as for the base detector.

3 The NAD Architecture

The NAD component was designed as highly modular software. The architectural model of NAD is presented in Fig. 1. High modularity eases maintaining software and cross-team development in potential future applications of the NAD component. Adding a new feature, e.g. new ML algorithm for building models, requires only implementation of the specific interface and does not require any knowledge about NAD. Additionally, in the process of component deployment, the user's role should be limited as much as possible, mainly to set system properties, which depend on the user's preferences (e.g. sensitivity of the detection algorithms) and to ensure normal behaviour of the monitored network and system during the learning phase. This section briefly describes the main modules of the component and possible deployment schemes.

3.1 Modules

According to the data flow within the component, the consecutively modules are as follow:

Features Extraction. Feature extraction is an essential part of the learning process. It transfers as many traffic characteristics as possible from raw traffic logs to numerical vectors (features values). On the other hand, the NAD development's main goal was to automate the component deployment from the user perspective, so feature extraction should be as a general process as possible. The NAD component contains a special tool (i.e. extensive features extractor) that makes implementing adaptors for any network type much more effortless.

The general concept of the feature extraction process is as follows. Let us assume that there is a set of messages. Each message has some message attributes (payload length, source IP address, gateway id, timestamp etc.). The list of attributes depends on the network type. We define **grouping operation** as dividing a set of messages into disjunctive groups by some criteria—grouping attributes (e.g. session-id divides traffic into TCP/IP flows, grouping by timestamp range divides traffic into time windows). Grouping operation can be hierarchical—set can be divided first by one grouping attribute (e.g. time)

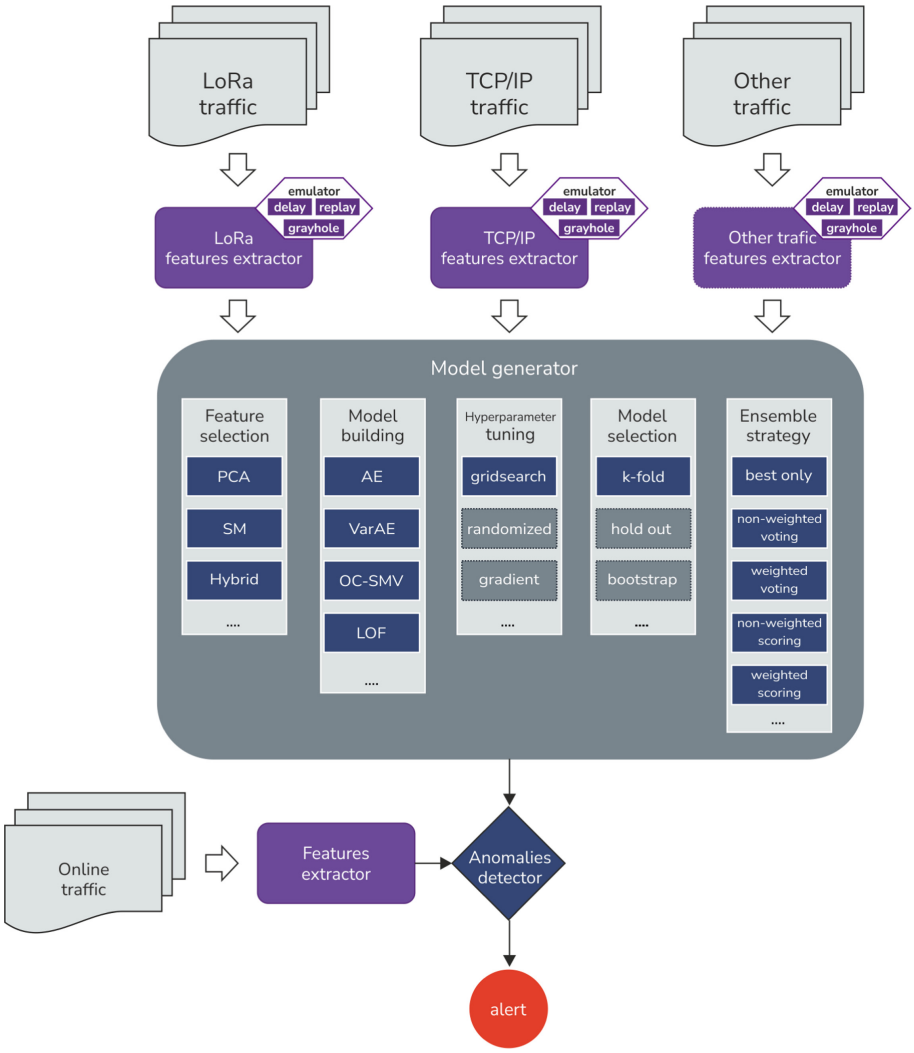


Fig. 1. The NAD modular architecture.

and then, e.g. into flows to obtain subgroups. When messages are grouped, each group can be characterised by a set of **group attributes**. The group attribute is a vector of values representing that group, e.g. intervals between consecutive messages in the group or payload length of each message. If the group attribute is not one of the message attributes (requires additional computation, e.g. intervals mentioned above), such attribute is called **derived attribute**.

As a result, each group is described with a set of vectors of different sizes, which is, in most cases, inconvenient for effective traffic modelling with ML. Hence, **aggregation functions** (e.g. sum, mean, max, min, skewness, etc.) are introduced to reduce the size of data. Each attribute can be aggregated with one or more functions. As a result of each aggregation, one value, called **feature value**, is obtained. If more than one grouping operation was applied, more aggregation operations also need to be involved (e.g. mean of means is calculated). Summing up, a **feature** is defined as a tuple of one or more grouping operations, one attribute and aggregation functions (number of aggregating functions is equal to the number of grouping operations). An exemplary feature definition is (time, source IP address, payload length, mean, mean). All computed features values create a vector that describes a set of messages used in further analysis.

Within the NAD component, versatile feature extraction [32] was implemented. The core idea of such feature extraction is to generate as many features as possible using many simple statistical measures as aggregation functions. The solution proved to be successful in domains with little domain knowledge. Little knowledge is also the case of unknown attack detection since it cannot be predicted which statistical measure will best reflect anomaly in network traffic during the specific attack. In NAD following aggregation functions were introduced:

- The mean value
- The maximal value
- The minimal value
- The range (difference between the maximal and minimal values)
- The sum of squared values (mean power)
- The standard deviation
- Skewness
- Kurtosis
- The 5th central moment
- The maximal difference between two consecutive measurements
- Autocorrelation taken at t = 1, 2, 5, 20 and 50
- Count of the given value (e.g. TCP in case protocol field)

Attack Emulation. The traffic is malformed within the feature extractor module: between raw traffic to standard format conversion and features extraction. The attack emulator contains a library of attacks (the current list includes DDoS, grayhole, replay, delay) and can be included in the new feature extractor "as it is". Each attack is configurable, e.g. intensity of the attack can be adjusted—performing the same type of attack with various configuration increases the quality of the abnormal dataset. Extending the library of attacks is possible if the specific attack for the given network type should be implemented.

Model Generator is the core component of NAD. The input data used in the model generator is defined as a set of samples—each sample (e.g. the flow in TCP/IP or traffic recorded in a time window) is represented by one vector. Within the model generator few submodules exist:

- Features selection—submodule responsible for selecting the most promising features of the network traffic (see Sect. 2.1).
- Model building—submodule responsible for building a model of the benign network traffic, mainly with the use of Machine Learning (see Sect. 2.1).
- Hyperparameters tuning—within the NAD platform, the space of hyperparameters values can be defined by the user. There exists several strategies for searching optimal set of hyperparameters values, e.g. grid search [26], randomized [8] and gradient [7].
- Model selection—various techniques can be used to validate the model quality and compare models to choose the most promising one. The exemplary of commonly used techniques are k-fold, bootstrap and hold out [17]. It should be stressed that in the NAD component, those techniques were modified due to introducing abnormal samples only to the validation set.
- Ensemble strategies—submodule responsible for combining multiple models into one to increase detection reliability (see Sect. 2.4).

Each submodule is easily extensible with new approaches.

Anomalies Detection. The model obtained from the model generator module examines traffic in the monitored network. Before the examination, the traffic is proceeded by the features extraction module mentioned above.

3.2 Deployment schemes

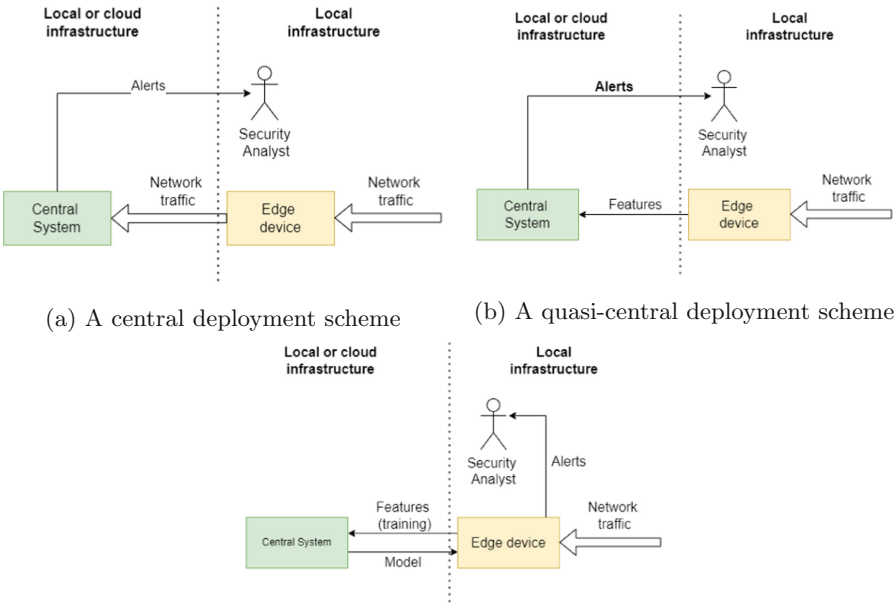
The high modularity of the architecture creates an opportunity to consider a few possible schemes of component deployment. In this section, we discuss three approaches (Fig. 2), but other variations are also possible.

Central Deployment (Fig. 2a). Network traffic is gathered in the edge device (e.g. router or gateway) and sent to the central system, where all modules of the NAD component are deployed. In this variant, resources of an edge device are barely used, but network load between edge and central nodes can be high, depending on the monitored network size and load.

Quasi-Central Deployment (Fig. 2b). The edge device is responsible not only for gathering traffic but also for feature extraction. The feature vectors are sent to the central node, and they're used there to train the model and examine new traffic. The network load is significantly lower than in the case of central deployment, whereas computation performed by edge devices does not require many resources. However, in the case of some resource-constrained devices, the burden of performing feature extraction can be significant.

Distributed Deployment (Fig. 2c). The edge device sends extracted features vectors to the central system in the training phase. Then the central system computes a model of the benign traffic and sends the model to the edge device. Then an edge device can examine network traffic without sending any network data outside of the system, which increases data safety. In this variant, the most resource-demanding computations are performed in the central system. Checking network traffic against the model of benign network traffic requires little computation. The drawback of this scheme is a potentially higher vulnerability for attacks, which use Adversarial Machine Learning techniques [15] due to a more accessible model of the benign traffic model for the attacker.

The choice of deployment scheme depends on the type of monitored network, resources available on the edge device, sensitivity of network data and risk of attacks based on adversarial machine learning.



(a) A central deployment scheme

(b) A quasi-central deployment scheme

(c) A distributed deployment scheme

Fig. 2. Data flows in various schemes of the NAD component deployment. The thickness of the lines indicates the size of the data.

3.3 Integration with the Real Network Infrastructure

The NAD component was integrated and successfully deployed within the GUARD project as one of the Security Services. GUARD is a cybersecurity framework to guarantee reliability and trust for digital service chains. One of the main goals of the GUARD structure is to improve the detection of attacks

and the identification of new threats, which the NAD component fits into. Two use cases were proposed as part of the project, and the appropriate testbeds were implemented. The NAD component was used to monitor LoRaWAN [28] network traffic within the use case related to the smart city [2]. In this use case, the NAD component was successfully integrated with the testbed and the real network deployed in Wolfsburg.

The GUARD framework architecture is presented in Fig. 3. The *GUARD agents* are deployed in monitored system infrastructure. Their responsibility is to gather security-related data from the monitored system and forward them to the core platform. There multiple *security services* are deployed. One of them is the NAD component. Due to GUARD framework architecture [3], the centralized variant of NAD deployment was chosen (Fig. 3). The component consumes network traffic from the ChirpStack Forwarder agent, which collects traffic from all LoRa gateways within the monitored system.

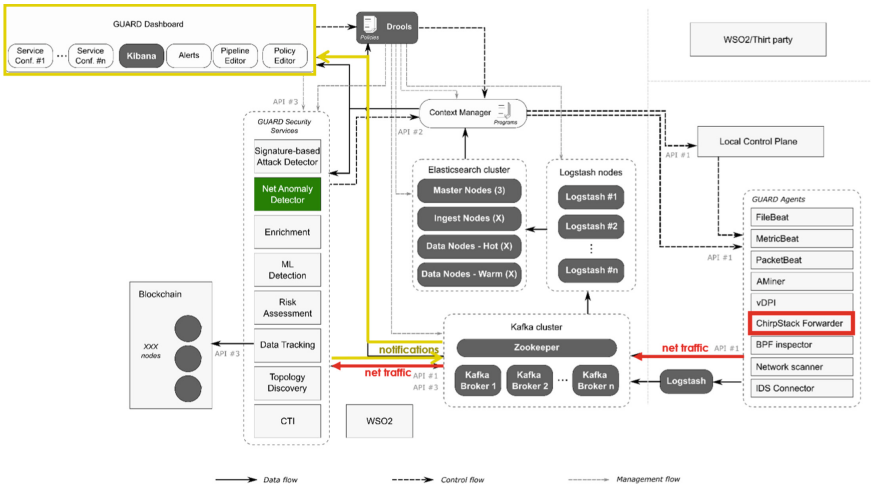


Fig. 3. GUARD architecture and data flow between GUARD components. With green color NAD component is marked. Arrows depict data flow. (Color figure online)

4 Experiments

In this work to conduct experiments on unknown attacks detection, the NAD component and the CIC-IDS2017 dataset [25] were used. This dataset contains TCP/IP traffic represented as a set of flows. Each flow is labelled according to the source—if it comes from a benign source or results from a malicious connection. In this experiment, three types of DDoS attacks are included in the validation set: DoS slowloris, DoS Slowhttptest and DoS Hulk. Only one attack, DoS Goldeneye, was included in the test dataset. In the case of flows analysis, anomaly detection aims not only to detect that the monitored network

is under attack (which in the case of DDoS attack is an easy task) but also to label flows responsible for the attack. As the model quality indicator and to tune hyperparameters values, the area under the precision-recall curve (AUC-PR) [27] measure was used. In the process of decision threshold adjustment, $f_{0.5}$ score was used to provide higher weight to the precision of the model over recall (in anomaly detection problems, the high false-positive rate is usually highly undesirable)

4.1 Repeatability

The main goal of the NAD component is to select the best configuration according to some given measure (AUC-PR in the case of experiments described in this section). In this process, all checked configurations are ranked. The ranking process needs to be characterised with high repeatability—the configurations should be ranked in the same or very similar order each time the process is repeated, independently on random factors (e.g. splitting the dataset into training, validation and testing sets). In this experiment k-fold approach, which is widely used in the ML area to compare decision models, is examined. The k-fold is slightly modified in our system as validation data is extended with abnormal traffic, which is not included in training data. Thus, some preliminary test of repeatability was performed on the exemplary dataset—we checked how much the ranking of configurations is similar within each configuration space. We selected five configuration spaces:

- Static feature selection + Autoencoder (AE, 9 configurations)
- Static feature selection + Variantional Autoencoder (VarAE, 9 configurations)
- PCA + SVM (PCA-SVM, 7 configurations)
- PCA + LOF (PCA-LOF, 42 configurations)
- Simple measures based feature selection + SVM (SM-SVM, 10 configurations)

The ranking method was applied three times on the same dataset (the learning and validation sets varied due to random split). Rank-biased Overlap (RBO) [31] measure was used to check the similarity of rankings. RBO measure can be parametrised with $p \in \langle 0, 1\rangle$ to determine the weight of the top ranks for the value of the similarity measure—$p = 1$ means that all ranks have equal weight, the smaller value of p the highest contribution of top ranks is. In the case of our approach, the most important is the repeatability of choosing a few top configurations (the best ones can be dropped due to instability; thus, the order of the following configurations is also important). In Table 1 RBO for $p = 1$ and $p = 0.5$ as well as standard deviation of APR AUC that determines the ranking are presented.

Table 1. Rank-biased Overlap of rankings obtained in three runs of the experiments and standard deviation of APR AUC of all configurations in the given space.

Configuration space	RBO ($p = 1$)	RBO ($p = 0.5$)	SD_{AUPRC}
AE	0.61	0.38	0.004
VarAE	0.85	0.64	0.006
PCA-SVM	0.86	0.58	0.001
PCA-LOF	0.93	0.97	0.115
SM-SVM	0.95	0.99	0.057

Table 2. The confusion matrix for SM-SVM detector (results for one run only).

		Predicted value	
		Negative	Positive
Actual value	Negative	0.64	0.006
	Positive	0.58	0.001

In the case of PCA-LOF and SM-SVM, the repeatability is high. For the rest configurations spaces, the repeatability of the process seemingly looks insufficient. However, the reason for low repeatability is a slight variance of the APR AUC within each space—the quality of all configurations is comparable: thus, even significant changes in ranking those configurations is irrelevant.

4.2 Unknown DoS Detection

The obtained base detectors (one for each configuration space) were examined against the test set (benign traffic and DoS Goldeneye, which was not seen previously by the system). The experiment was repeated three times as before. The $f_{0.5}$-score value obtained by each base detector on the validation set (known attacks) and the test set (unknown attack) is shown in Fig. 4. Firstly, it can be observed that the quality of detection for the test set is lower (except SM-SVM). The high standard deviation of PCA-SVM was caused by choosing another configuration in one of three experiment runs. As expected, that configuration performed well on the validation set ($f_{0.5}$-score equal to 0.92), similarly to all other configurations in this configuration space (see Table 1). However, surprisingly it was significantly weaker on the test set ($f_{0.5}$-score equal to 0.59).

The confusion matrix for the first run of the experiment for the SM-SVM detector (the best detector according to the mean score) is presented in Table 2. The frequency of raised alarms during the attack is much higher than when the system operates in normal conditions, making detection that the monitored system is under attack easy. Additionally, as intended, the FPR is relatively low. However, in case of mitigation of the attack (e.g. by dropping the suspicious traffic), the number of dropped benign connections may be unacceptable.

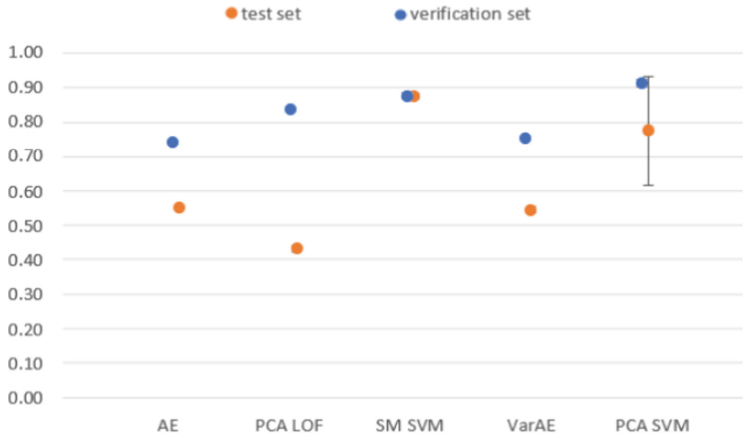


Fig. 4. Average $f_{0.5}$-score and standard deviation obtained by base detection models for the validation set (in blue) and the test set (in orange). (Color figure online)

4.3 Ensemble Models

The weight of each base detector was calculated based on the value of the APR AUC measure (see Table 3) obtained on the validation set. In each experiment run, the PCA-SVM detector was recognized as the best detector (the highest value of APR AUC on the validation set). The quality of all ensemble detectors for the testing set is shown in Fig. 5. It can be observed that the high standard deviation of the most significant detector (PCA-SVM) propagates on ensemble detectors, except for the case of weighted voting strategy. In the case of all assemble detectors, the standard deviation of the results is smaller than for the PCA-SVM base detector. In this scenario, the weighted voting strategy proved to be insensitive to the instability of base detectors, increasing the robustness of the approach.

Table 3. Quality (APR AUC) and weight of base detectors obtained on the validation set.

Base detector	APR AUC		Weight
	Mean	Standard deviation	
AE	0.82	0.001	0.19
VarAE	0.81	0.002	0.19
PCA-SVM	0.92	0.007	0.22
PCA-LOF	0.80	0.007	0.19
SM-SVM	0.91	0.003	0.21

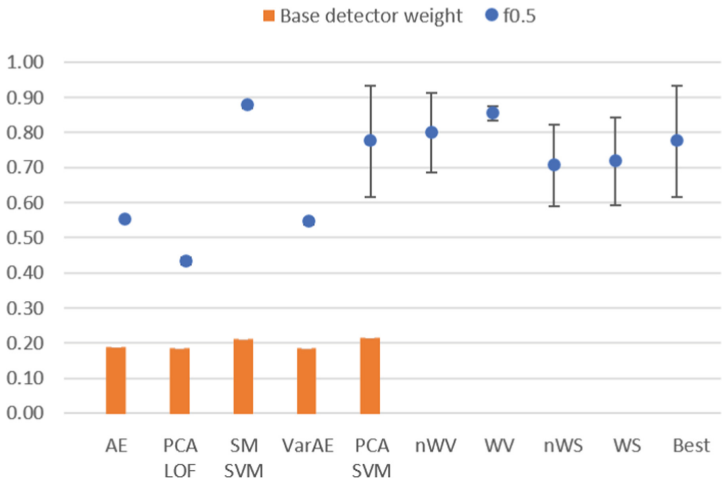


Fig. 5. Base detectors weights (marked with orange), the mean and standard deviation of base and ensemble detectors (marked with blue). (Color figure online)

4.4 Threshold Adjustment

The performance of each detector depends on how well the model evaluates the anomaly level of the sample (scoring) and on the detection threshold value—the minimal score value to assess the sample as an anomaly. The threshold in NAD is calculated based on the validation set that contains initial traffic and exemplary malicious traffic (originated from known attacks), which may be much different from the malicious traffic in the test set (derived from unknown attacks). The experiment aimed to check how much optimal threshold value acquired with the validation set differs from the one calculated for the test set and how this difference influences the model quality.

In our experiment, $f_{0.5}$-score measure is used to calculate the optimal value of the threshold. In Figs. 6 and 7, the precision-recall curves for the weighted votes detector and Autoencoder detector for the test set are presented, respectively. In both figures, green dots denote the optimal threshold values calculated for the validation set, while red dots denote the optimal threshold values calculated for the test set. Although the curves differ, their area is similar—the WV detector performs better if a small recall is preferable, the AE in the opposite situation. In the case of the WV detector, the value of the threshold obtained with the validation set is close to the optimal value for the test set, which is a highly desirable situation. In the case of the Autoencoder detector, the optimal value of the threshold obtained with the validation set does not perform well in the case of the test set. The AE model quality makes it possible to obtain $f_{0.5}$-score equal to 0.8 (if optimal threshold value could be calculated on the test set). Thus, it may be concluded that the quality of the threshold adjustment is mainly responsible for the poor detector performance on the test set ($f_{0.5}$-score equal to 0.55). On the other hand, the threshold adjustment mechanism performs well for the WV detector and other ensemble detectors.

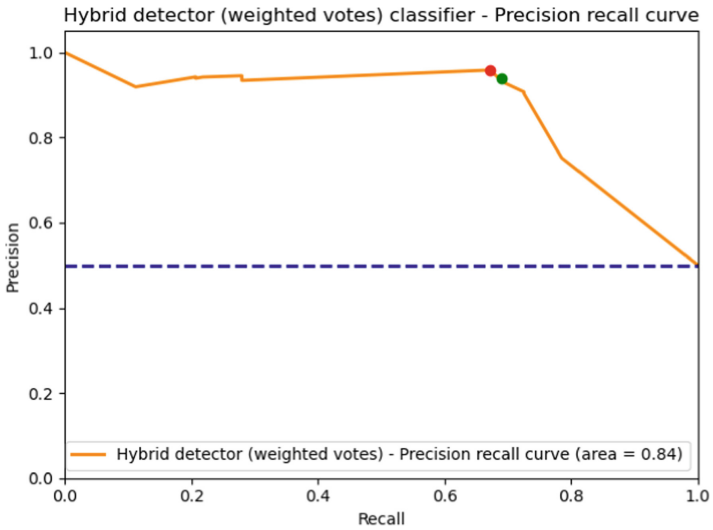


Fig. 6. Precision recall curve for Weighted Votes detector for test set. Green dot denotes the optimal threshold value calculated for the validation set, while red dote denotes the optimal threshold value calculated for the test set. (Color figure online)

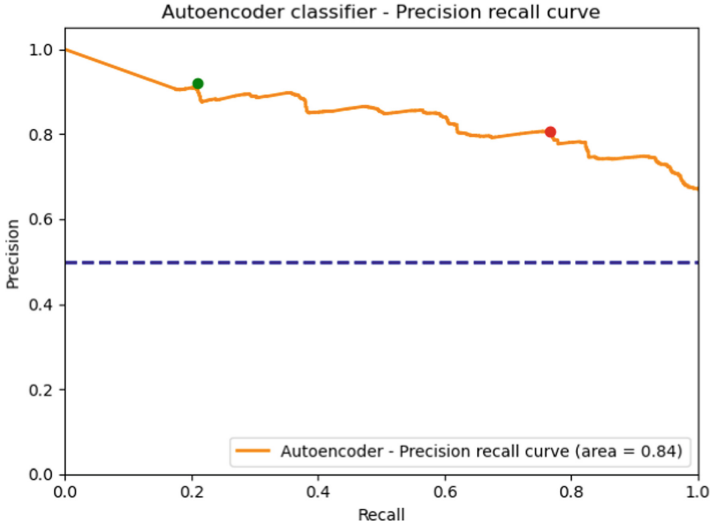


Fig. 7. Precision recall curve for Autoencoder detector for test set. Green dot denotes the optimal threshold value calculated for the validation set, while red dote denotes the optimal threshold value calculated for the test set. (Color figure online)

5 Conclusions

In this chapter, we presented the Net Anomaly Detector component, which is designed to detect anomalies in network traffic of various types using ML. The component incorporates several mechanisms for increasing the quality of unknown attack detection, i.e. known attacks emulation, exhaustive feature extraction, hyperparameter tuning, detection threshold adaptation and ensemble models strategies. The architecture of the component, its potential deployment variances and exemplary deployment within the GUARD framework were presented. We showed that: the proposed hyperparameter tuning and modified k-fold are characterised with acceptable repeatability; adjusting hyperparameters values and detection threshold values with the use of dataset with samples malformed with known attacks allows to detect of unknown attacks with high quality; application of some ensemble strategies increase the robustness of the anomalies detection. However, the presented results are preliminary and more experiments on heterogeneous datasets are required.

Acknowledgement. This work was supported in part by the European Commission under Grant Agreement no. 833456 (GUARD).

References

1. Check point's 2021 cyber security report reveals extent of global cyber pandemic, and shows how organizations can develop immunity. https://pages.checkpoint.com/cyber-security-report-2021.html. Accessed 10 Jan 2022
2. D2.1 vision, state of the art and requirements analysis. https://guard-project.eu/wp-content/uploads/2019/11/GUARD_D2.1_Vision-State-of-the-Art-and-Requirements-Analysis.pdf. Accessed 10 Jan 2022
3. D2.2 guard reference architecture. https://guard-project.eu/wp-content/uploads/2020/04/D2.2_GUARD-Reference-Architecture.pdf. Accessed 10 Jan 2022
4. Ahmed, M., Mahmood, A.N., Hu, J.: A survey of network anomaly detection techniques. J. Netw. Comput. Appl. **60**, 19–31 (2016)
5. Amer, M., Goldstein, M.: Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In: Proceedings of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012), pp. 1–12 (2012)
6. An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE **2**(1), 1–18 (2015)
7. Bengio, Y.: Gradient-based optimization of hyperparameters. Neural Comput. **12**(8), 1889–1900 (2000)
8. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2), 281–305 (2012)
9. Bewick, V., Cheek, L., Ball, J.: Statistics review 13: receiver operating characteristic curves. Critic. Care **8**(6), 1–5 (2004)
10. Bhatti, M.A., Riaz, R., Rizvi, S.S., Shokat, S., Riaz, F., Kwon, S.J.: Outlier detection in indoor localization and internet of things (IoT) using machine learning. J. Commun. Netw. **22**(3), 236–243 (2020)

11. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 93–104 (2000)
12. Chen, Z., Yeo, C.K., Lee, B.S., Lau, C.T.: Autoencoder-based network anomaly detection. In: 2018 Wireless Telecommunications Symposium (WTS), pp. 1–5. IEEE (2018)
13. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
14. Dietterich, T.G., et al.: Ensemble learning. The handbook of brain theory and neural networks. Arbib MA **2**(1), 110–125 (2002)
15. Grieco, G., Grinblat, G.L., Uzal, L., Rawat, S., Feist, J., Mounier, L.: Toward large-scale vulnerability discovery using machine learning. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 85–96 (2016)
16. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems, vol. 18. MIT Press (2006). https://proceedings.neurips.cc/paper/2005/file/b5b03f06271f8917685d14cea7c6c50a-Paper.pdf
17. Kim, J.H.: Estimating classification error rate: repeated cross-validation, repeated hold-out and bootstrap. Comput. Stat. Data Anal. **53**(11), 3735–3745 (2009)
18. Lane, T., Brodley, C.E.: An application of machine learning to anomaly detection. In: Proceedings of the 20th National Information Systems Security Conference, vol. 377, pp. 366–380. Baltimore, USA (1997)
19. Lorena, L.H., Carvalho, A.C., Lorena, A.C.: Filter feature selection for one-class classification. J. Intell. Robot. Syst. **80**(1), 227–243 (2015)
20. Moya, M.M., Hush, D.R.: Network constraints and multi-objective optimization for one-class classification. Neural Netw. **9**(3), 463–474 (1996)
21. Nassif, A.B., Talib, M.A., Nasir, Q., Dakalbab, F.M.: Machine learning for anomaly detection: a systematic review. IEEE Access **9**, 78658–78700 (2021)
22. Partridge, M., Calvo, R.A.: Fast dimensionality reduction and simple PCA. Intell. Data Anal. **2**(3), 203–214 (1998)
23. Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Process. **99**, 215–249 (2014)
24. Reilly, B.: Social choice in the south seas: electoral innovation and the borda count in the pacific island countries. Int. Politic. Sci. Rev. **23**(4), 355–372 (2002)
25. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSP **1**, 108–116 (2018)
26. Shekar, B.H., Dagnew, G.: Grid search-based hyperparameter tuning and classification of microarray cancer data. In: 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), pp. 1–8 (2019). https://doi.org/10.1109/ICACCP.2019.8882943
27. Sofaer, H.R., Hoeting, J.A., Jarnevich, C.S.: The area under the precision-recall curve as a performance metric for rare binary events. Method. Ecol. Evol. **10**(4), 565–577 (2019)
28. Sornin, N., Yegin, A.: Lorawan™ 1.1 specification (2017). https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf. Accessed 10 Jan 2022
29. Tang, Y., Tong, Q.: Bordarank: A ranking aggregation based approach to collaborative filtering. In: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), pp. 1–6 (2016). https://doi.org/10.1109/ICIS.2016.7550761

30. Wang, Y., Wong, J., Miner, A.: Anomaly intrusion detection using one class SVM. In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, pp. 358–364 (2004). https://doi.org/10.1109/IAW.2004.1437839
31. Webber, W., Moffat, A., Zobel, J.: A similarity measure for indefinite rankings. ACM Trans. Inf. Syst. **28**(4), 1–38 (2010). https://doi.org/10.1145/1852102.1852106
32. Zagorecki, A.: A versatile approach to classification of multivariate time series data. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 407–410. IEEE (2015)