IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 January 2024, accepted 14 February 2024, date of publication 20 February 2024, date of current version 29 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3368377

RESEARCH ARTICLE

Explainable AI for Intrusion Detection Systems: LIME and SHAP Applicability on Multi-Layer Perceptron

DIOGO GASPAR[1], PAULO SILVA[1,2], AND CATARINA SILVA[2], (Senior Member, IEEE)
[1]Laboratório de Informática e Sistemas (LIS), Instituto Pedro Nunes, 3030-199 Coimbra, Portugal
[2]CISUC/LASI – Centre for Informatics and Systems, Department of Informatics Engineering, University of Coimbra, 3004-512 Coimbra, Portugal

Corresponding author: Diogo Gaspar (jgaspar@ipn.pt)

This work was supported in part by the Autonomous Trust, Security and Privacy Management Framework for Internet of Things (IoT) (ARCADIAN-IoT), under Grant 101020259(H2020-SU-DS02-2020); in part by the National Funds through the Foundation for Science and Technology (FCT), I.P., within the Scope of the Project Centro de Informática e Sistemas da Universidade de Coimbra (CISUC), under Grant UID/CEC/00326/2020; and in part by the European Social Fund through the Regional Operational Program Centro, in 2020.

ABSTRACT Machine learning-based systems have presented increasing learning performance, in a wide variety of tasks. However, the problem with some state-of-the-art models is their lack of transparency, trustworthiness, and explainability. To address this problem, eXplainable Artificial Intelligence (XAI) appeared. It is a research field that aims to make black-box models more understandable to humans. The research on this topic has increased in recent years, and many methods, such as LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations) have been proposed. Machine learning-based Intrusion Detection Systems (IDS) are one of the many application domains of XAI. However, most of the works about model interpretation focus on other fields, like computer vision, natural language processing, biology, healthcare, etc. This poses a challenge for cybersecurity professionals tasked with analyzing IDS results, thereby impeding their capacity to make informed decisions. In an attempt to address this problem, we have selected two XAI methods, LIME, and SHAP. Using the methods, we have retrieved explanations for the results of a black-box model, part of an IDS solution that performs intrusion detection on IoT devices, increasing its interpretability. In order to validate the explanations, we carried out a perturbation analysis where we tried to obtain a different classification based on the features present in the explanations. With the explanations and the perturbation analysis we were able to draw conclusions about the negative impact of particular features on the model results when present in the input data, making it easier for cybersecurity experts when analyzing the model results and it serves as an aid to the continuous improvement the model. The perturbations also serve as a comparison of performance between LIME and SHAP. To evaluate the degree of interpretability increase, and the explanations provided by each XAI method of the model and directly compare the XAI methods, we have performed a survey analysis.

INDEX TERMS Artificial intelligence, explainability, intrusion detection system, local interpretable model-agnostic explanations, machine learning, shapley additive explanations.

I. INTRODUCTION

Nowadays, network security is extremely important. It is estimated that by this year will be a trillion physical devices connected to the internet [1]. To cybersecurity, this is a big concern. Devices connected to the Internet with wide distribution and openness are the ideal targets for cyber attacks. Since new equipment with this characteristics is being made and connected every day, collecting and processing private information, they become the perfect targets for attackers [2].

In recent years, the field of Artificial Intelligence (AI) has seen significant advancements in terms of learning performance of Deep Learning (DL) methods, compared

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

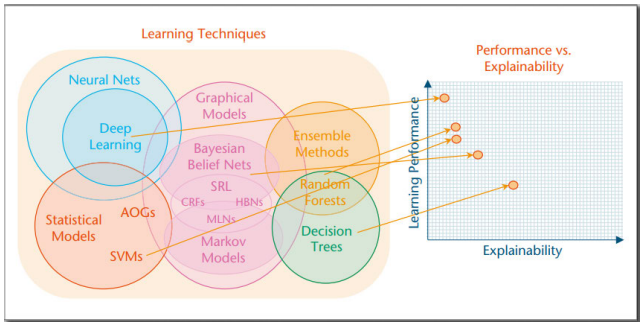


FIGURE 1. Explainability of machine learning models appear inverse to their prediction accuracy [3].

to traditional Machine Learning (ML) methods. Intrusion Detection Systems (IDS) have proven to be effective, especially with the use of Deep Neural Networks (DNNs) that have improved the detection rates of such models. However, the problem with some state-of-the-art models is their complexity, making them hard to understand when analyzed by humans.

There are models considered interpretable, e.g., Decision Trees, Decision Rules and Linear Models. Unfortunately, usually there is a tradeoff between explainability of the model and its learning performance. Explainability of a machine learning model is usually inverse to its learning performance [3]. Models with high learning performance are not as explainable as simpler models as Figure 1 shows. We can see so in the figure that Decision Trees have high explainability values. However, looking at the learning performance, they are the learning techniques that present the lowest learning prediction. In contrast, Deep Learning techniques such as Deep Neural Networks (DNNs) show high values of learning performance but are the least explainable. The motivation for eXplainable Artificial Intelligence (XAI) research is to increase the explainability of these high learning performance models.

eXplainable Artificial Intelligence (XAI) is a research field that aims to address the lack os explainability and trusworthiness of black-box models and make them more understandable to humans. The research on this topic has increased in recent years. However, the majority of work and research about interpretation and transparency of models is focused on other fields like computer vision, natural language processing, biology, and healthcare making model interpretability on IDS still very uncovered.

In non-critical applications such as product recommendations on e-commerce websites or friend suggestions on social networks, a wrong prediction may not have severe consequences. However, in critical applications such IDS or medical diagnosis, the lack of transparency, trustworthiness, and explainability in these models poses significant risks. Thus, there is a need for techniques that enhance the interpretability and increase trustworthiness on these models. For those reasons, in this work, we used two XAI methods such as LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations) to provide explanations and increase the interpretability of a black box model part of an IDS solution that performs intrusion detection on IoT devices.

In the execution of our work we have used a generated version of the ADFA-LD dataset. The model receives sequences of system calls - a programmatic request for a service provided by the operating system, enabling applications to interact with the underlying hardware and access various system resources. The model output is 0 if the sequence is deemed normal and 1 if an anomaly is detected. Given that analyzing system call sequences when unexpected events occur is a complex task, providing explanations about the model predictions gives cybersecurity personnel insights into the model's behavior, enabling them to make informed decisions about the results. In our approach, we have additionally conducted a perturbation analysis based on the explanations to obtain a different input, thereby validating the explanations. Through perturbation analysis, the adverse impact of specific system calls on classification, when present in the input, is identified. This serves as added value when scrutinizing the model and its results, and it proves advantageous for the continuous improvement of the model. As an additional step, we have conducted a survey analysis in which participants answered questions evaluating the degree of interpretability increase when each XAI method is used, directly comparing both methods. Additionally, it assesses the level of necessity for XAI methods and interpretability in the analysis of black-box models. This work also leaves open the possibility of being enhanced applying this approach to other models and types of data.

The main contributions of this work are the following:

(i) We propose an architecture that uses XAI methods such as LIME and SHAP to provide local explanations for the results of a black-box model that is part of an IDS solution that performs intrusion detection on IoT devices. This architecture is beneficial to the understanding of the predictions made from the IDS, and help cybersecurity personnel to interpret the those predictions with insights of the model behavior.

(ii) In addition to the local explanations, we use an approach to validate and enhance the trust on the explanations, based on a perturbation analysis that takes into consideration the explanations provided. The input suffers perturbations in an attempt to obtain a different output, hence validating the explanations provided. Since we use two XAI methods, the perturbation analysis can also serve as a comparison of the XAI methods performance.

(iii) To evaluate the explanations provided from the XAI methods, and make a direct comparison between the methods, we carried out a survey where participants answered questions destined to evaluate the degree of interpretability increase of the model using each method, and answered questions that directly compared the XAI methods, choosing the preferred method.

(iv) During the experimental phase, we successfully identified features with adverse effects on the model outcome

when present in the input. This holds paramount significance for cybersecurity professionals, as it allows for informed analysis of model results. The recognition of these features in the input signals potential inaccuracies in the output. It is notable to highlight its importance in ensuring the continuous improvement of the model.

The rest of the paper is organized as follows. Section II describes the related works. The background is discussed in section III. In section IV we present our proposed approach to increase the interpretability of a black box model part of an IDS solution for IoT devices, and the validation and evaluations steps of the explanations provided. In section V we describe the experiments carried out using a generated version of the ADFA-LD dataset, presenting the results in detail. Finally, section VI presents the conclusion to this work and how it could be further enhanced.

II. RELATED WORK

Although XAI is a captivating research area, the majority of research efforts are directed towards domains like healthcare, natural language processing, and computer vision. The interpretability methods are rarely used for intrusion detection. However, with the intrusion detection systems becoming more complex, works to improve the interpretability of such models have emerged.

Wang et al. [4] propose a framework that leads to improve the transparency of any IDS, presenting the first use of SHAP to provide local and global explanations for intrusion detection. The authors interpret and compare two different classifiers, one-vs-all classifier and multiclass classifier using the NSL-KDD dataset to test the framework feasibility. The experimental results show that the interpretation results, generated by the framework, are consistent with the characteristics of the specific attacks, and the results are very intuitive.

Patil et al. [5] proposes an innovative intrusion detection system, using ensemble methods of machine learning and incorporating the XAI method LIME for better explainability and understanding of the black-box approach to reliable intrusion detection. The experimental results confirm LIME is more explanation-friendly and more responsive.

Marino et al. [6] presented an approach to generate explanations for incorrect classifications made by data-driven IDS. An adversarial approach is used to find the minimum modifications (of the input features) required to correctly classify a given set of misclassified samples. The magnitude of the modifications is used to visualize the most relevant features that explain the reason for the misclassification. Experimental evaluation was conducted on the NSL-KDD99 benchmark dataset using Linear and Multilayer perceptron classifiers.

Mane and Rao [7] have used deep neural network for network intrusion detection and also proposed explainable AI framework to add transparency at every stage of machine learning pipeline. The explanations are generated from SHAP, LIME, Contrastive Explanations Method (CEM), ProtoDash and Boolean Decision Rules via Column Generation (BRCG). The approaches were applied to NSL-KDD dataset for intrusion detection system (IDS).

Siganos et al. [8] introduced an AI-powered IDS with explainability functions for the IoT. The proposed IDS relies on ML and DL methods, while SHAP is used to explain decision-making. The evaluations results demonstrate the efficiency of the proposed IDS in terms of detections performance and explainable AI (XAI).

Barnard et al. [9] present a two-stage pipeline for robust network intrusion detection. The authors implement an extreme gradient boosting (XGBoost) model to perform supervised intrusion detection, and leverage the use of SHAP to provide explanations. In a second stage, the explanations are used to train an auto-encoder to distinguish between previously seen and unseen attacks. Experiments were conducted on the NSL-KDD dataset showing that the solution accurately detect new attacks encountered during testing, while the overall performance is comparable to numerous state-of-the-art works from literature.

Neupane et al. [10] conducted a comprehensive survey on Explainable Intrusion Detection Systems (X-IDS), emphasizing the importance of transparency and interpretability in the context of evolving cyber threats. The paper provides an overview of current methods, challenges, and opportunities in X-IDS, making it a valuable resource for researchers and practitioners in the field.

Sharma et al. [11] propose a deep learning-based approach for intrusion detection in IoT networks. They address security issues in IoT layers and use XAI techniques such as LIME and SHAP to interpret the deep learning model's decisions. The approach achieves high accuracy and offers valuable insights for researchers in this field.

Even though the past works use XAI methods for intrusion detection, they focus on the part of developing an IDS and apply the XAI methods to have explanations, either local or global in scope. Although there has been adoption of XAI methods like LIME and SHAP, the literature lacks validation, evaluation, and comparison of the results of such methods. These steps are important not only to ensure the reliability and robustness of these methods but also for facilitating meaningful comparisons between different XAI techniques, allowing possible end-user of such techniques to make well-informed decisions when it comes to the selection and use of XAI methods. In this work, we went further and address it; we explored the explanations provided by LIME [12] and SHAP [13] through evaluation and validation analysis.

In the validation analysis, perturbation on the input data was performed based on the explanations in order to obtain a different output to attempt to check the truthfulness of the explanations. Through the validation analysis we were also able to retrieve system calls that have a negative impact on the classification of the model when present in the input that will be discussed in more detail on Section V. The evaluation analysis was based on the validation and on a

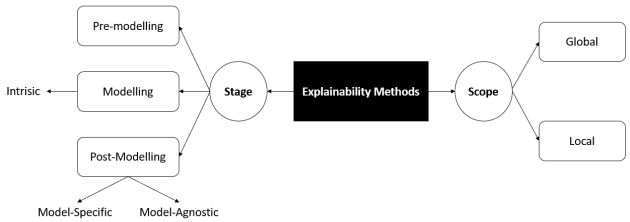


FIGURE 2. A taxonomy of eXplainable AI methods.

survey where the participants answered questions on how the explanations are important for a better understanding of the model. In the survey, a direct comparison of both XAI methods used (LIME and SHAP) is performed to evaluate which method the participants felt more confident with.

We have selected LIME and SHAP due to their model-agnostic nature, allowing their application to multiple models and data types, also offering local explanations. The selection of these methods was also influenced by their widespread use in the literature, and we will elaborate more on them in the next section. However, there are other well-known and used XAI methods that could have been selected, such as Anchors [14], Accumulated Local Effects (ALE) [15], Counterfactual Explanations [16], and Contrastive Explanations Method (CEM) [17]. For further improvement of this work such methods could be considered to be applied on the approach and compared with LIME and SHAP.

III. BACKGROUND

There are many factors that contribute to how a model operates and makes its predictions and, therefore, many ways to explain it. XAI methods can be classified using two dimensions: stage and scope.

The extent of the explainability method can vary, with the method capable of offering either a global explanation encompassing the entire model or a local explanation specific to a single prediction, or both.

A taxonomy of XAI methods can be seen in Figure 2. The explainability method can be applied through the three stages of an AI development model (pre-modelling, modelling, and post-modelling). The two main stages where explainability can be applied are the modelling (intrinsic explainability), and the post-modelling (post-hoc explainability) [18]. Intrinsic models refer to interpretable models like decision trees or rule-based models. Post-hoc models involve extracting information from pre-trained models, and their effectiveness is not tied to the inner workings of the model, making them either model-agnostic or model-specific when the XAI method is developed specific to a DL model.

A. LIME

The authors of LIME [12] propose a concrete implementation of local surrogate models. Local surrogate models explain individual predictions of a black-box model. LIME focuses on training local surrogate models to explain individual predictions. LIME is based on a perturbation mechanism, testing the model with variations of the data. Generates a new dataset consisting of permuted samples and the corresponding predictions of the model. On this new dataset LIME trains an interpretable model, weighted by the proximity of the sampled instances to the instance of interest. The local surrogate model should be a good approximation of the black box model predictions locally. The local surrogate model can be calculated in the following way:

$$\xi(x) = \arg\min_{g \in G} L(f, g, w^x) + \Omega(g), \tag{1}$$

where - g represents the explanation model for the instance x, (e.g., decision tree). G is the family of possible explanations. For example all possible decision tree models. L is the loss function (e.g., mean squared error), which is used to measure how close the predictions from the explanation model are to the original. f represents the original model. w^x defines the wight between the sampled data and the original data. If the sampled data is similar to the original data, the weight is greater, and vice versa. Ω ((g) represents the complexity of model g. Steps must be followed [19] to train a surrogate model:

1) Select instance of interest to have an explanation for its black-box prediction;
2) Perturb dataset and get predictions of the new points;
3) Weight the new samples according to their proximity to the instance of interest;
4) Train a weighted, interpretable model on the dataset with the variations;
5) Explain the prediction by interpreting the local model.

B. SHAP

SHAP is an unified framework proposed by Lundberg and Lee [13] for interpreting predictions. Assigns each feature an importance value for a particular prediction and connects LIME and Shapley values.

One innovation that SHAP has is that the Shapley value explanation is represented as a linear model. That view connects LIME and Shapley Values. Each SHAP value measures how much each feature in the model contributes, either positively or negatively. It offers two essential benefits: it can be calculated for any model rather than just simple, linear models and each record has its own set of SHAP value. SHAP values provide three significant advantages compared to other methods. First, SHAP has a solid theoretical foundation in game theory [4]. Shapley values are the only attribution method to satisfy the properties: Efficiency, Symmetry, Dummy, and Additivity [19]. Efficiency refers to the feature contributions that must add up to the difference of prediction for x and the average. Symmetry is about the contributions of two feature values j and k should be the same if they contribute equally to all possible coalitions. Dummy refers to a feature j that does not change the predicted value, regardless of which coalition of feature values it is added to

and should have a Shapley value of 0. Additivity means that the total Shapley value assigned to a coalition of players or features is equal to the sum of the individual Shapley values of the players or features when they join the coalition. SHAP can also satisfy these since it gets Shapley values from linear models. Second, SHAP connects LIME and Shapley values. It helps to unify the field of interpretable machine learning. At last, SHAP has a fast computation for machine learning models compared to calculating Shapley values directly. SHAP specifies the explanation for an instance x [4] as:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j, \tag{2}$$

where - g is the explanation model. z' is the coalition vector (also called simplified features), and $z' \in 0, 1^M$. The 1 in z' means the features in the new data are the same as those of the original data (the instance x), while the 0 means the features in the new data are different from those of the original (the instance x). M is the maximum coalition size. $\phi_j \in \mathbb{R}$ is the feature attribution for the feature j for instance x. It is the Shapley value. If ϕ_j is a large positive number, it means feature j has a large positive impact on the prediction made by the model.

IV. PROPOSED APPROACH

A. RETRIEVAL OF EXPLANATIONS

To implement our approach, we applied the selected XAI methods (LIME and SHAP) to a black-box model integrated into a solution designed for intrusion detection in IoT devices. Both methods exhibit model-agnostic characteristics, accommodating multiple models and data types, and offer local explanations. The selection of these methods was influenced by their widespread use in the literature and their similarity, a crucial factor facilitating a meaningful comparison between them. However, some changes on the methods were necessary in order to achieve the desired setup. The dataset used is a generated version of the ADFA-LD [20], [21], [22] dataset that we will further describe with more detail in section V. The dataset is based on system calls, with normal traces and attack traces with attacks such as password brute force, denial of service, and network scan.

For the explanations with LIME, we used the LimeTabularExplainer [23], where we defined our explainer and fed it information about the model and data. We provided to the explainer the data, the feature names, and the class names. For every dataset instance, explanations were retrieved. It was defined that the explanations would be made using the 10 most important features of each instance.

In the execution of the explanations with the second XAI method (SHAP), the approach is similar. As in the LIME case, SHAP supports numerous models and data types. For our approach we decided to use the KernelExplainer [24] as it is the model-agnostic explainer of SHAP. For all the instances, as in LIME, the explanations were retrieved using the 10 most important features of each instance.

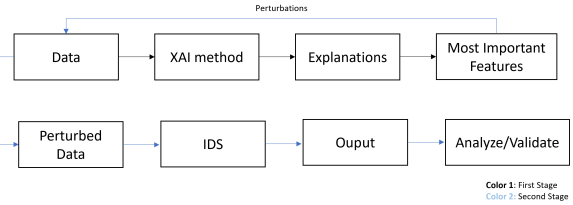


FIGURE 3. Perturbation analysis experiments base architecture.

At the conclusion of this step, we obtain explanations from LIME for all instances in the dataset, along with corresponding explanations from SHAP for the same instances in the dataset.

B. EVALUATION OF EXPLANATIONS

In order to better analyze, validate, and evaluate the explanations provided by the XAI methods, we have performed a perturbation analysis based on the explanations.

Perturbation analysis is a valuable technique used to understand the behaviour of the system when subjected to small changes or disturbances in its inputs. The primary goal of the perturbation analysis is to assess how the model responds to the small perturbations carried out. First, we have collected the 10 most important features for the classification of each instance (as presented in the subsection above). Then we have performed perturbations on the input data of each instance by perturbing the 10 most important features previously collected. The perturbations were based on removing/replacing these features from the original input instances, generating new input instances. After the perturbations, the new generated input instances were fed back to the model with the goal of obtaining different outputs, validating and evaluating the explanations provided by the XAI methods. Figure 3 shows the main layout of the experiments that we just described. The results, and experiments are explained in more detail in Section V.

For further validation and evaluation of the XAI methods and explanations provided by them, we have performed a survey analysis. The participants of the survey had to answer questions about the degree of interpretability increase by each method, which method they felt more confident with/preferred and why for the exact same problems, if they felt that XAI is important to help the interpretability of black box models, and other questions to compare the XAI methods, and evaluate their explanations. The survey and questions and answers as well as information regarding the participants is present further on Section V.

C. ALGORITHMS

In this section we explain in more detail the algorithms used. The base architecture can be seen in Figure 3. For an overall analysis we have proceeded with the first algorithm presented in detail in Figure 4. We start by collecting the explanations for all the instances of the dataset, using LIME and SHAP, saving the explanations of each separately. Through the

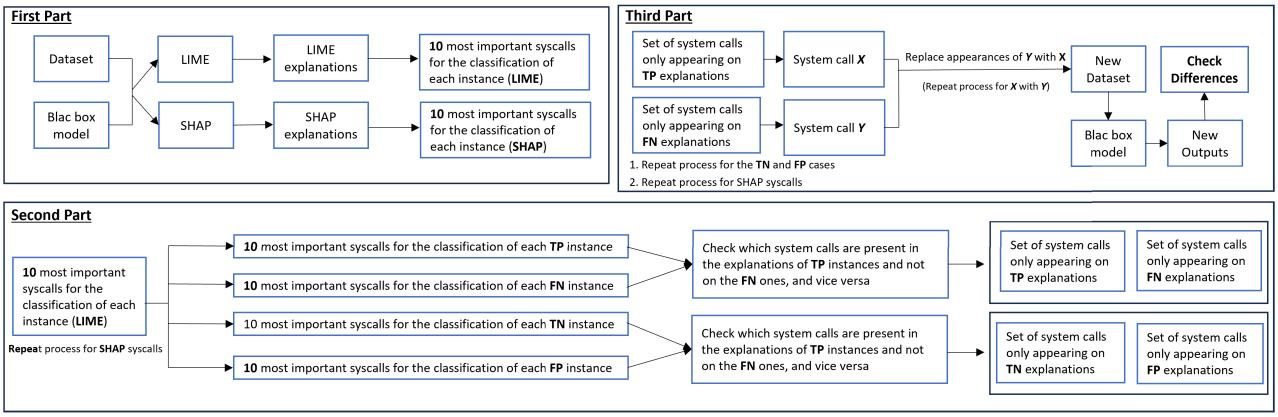


FIGURE 4. First algorithm scheme for the overall perturbation analysis.

explanations of each method, we are able to retrieve the 10 most important system calls for the classifications of each instance and save them separately by the ones provided from LIME and the ondes from SHAP. This is the first part. For the second part, we use the same approach for both LIME

and SHAP, but separately also for each. We start by dividing the collected most important system calls by true positive instances, false negatives, true negatives, and false positives. Then we check which and if are there system calls present in the explanations of TP cases that are not present in the explanations of FN (opposite) cases. If so we save them and we perform the same for the opposite and repeat the process for the TN and FP cases. This being, we have a set of system calls that only appear in the explanations of TP and not in FN explanations and a set for the opposite. We also have two sets for the TN and FP the same way. For the third step, we select a system call of the set of TP (X), and one of the FN (Y). We then proceed to replace in the dataset, the appearances of Y with X generating a new modified dataset that we feed back to the black box model and observe the changes in the output. We repeat the process for all the system calls, combining all possible X with all possible Y. We repeat the process for the TN and FP cases.

For the first batch of specific experiments, we simply remove from each instance of the dataset, the 10 most important system calls, result from the XAI methods explanations, for the classification of each instance, generating a new modified dataset. This new modified dataset differs from the original by having 10 system calls removed from each instance. We then feed the new dataset to the model and check the differences in the output. The algorithm is presented in Figure 5.

The second and third batch of experiments are similar. On the second batch we start by iteratively select a system call (X) from the set of system calls only appearing on the explanations of TP cases, and one (Y) from the FN cases. Then, on every FN instance, we replace the 10 most important system calls for the classification of that instance for X, and on the instances of TP, the same but for Y. Then we feed the new generated instances back to the model

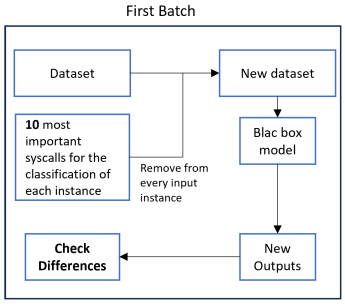


FIGURE 5. Algorithm for the first batch of experiments.

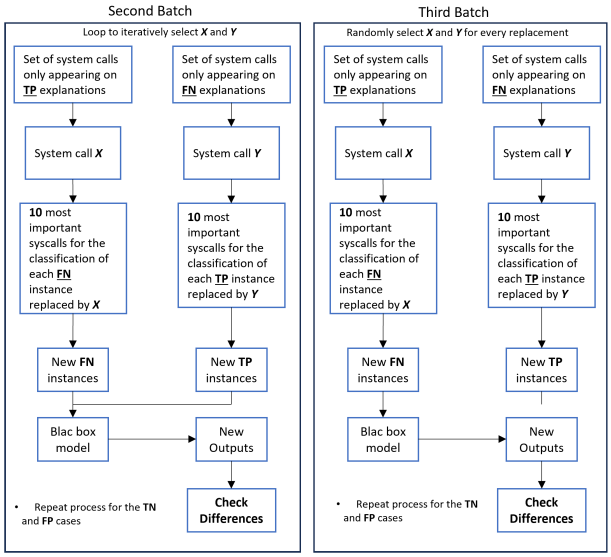


FIGURE 6. Algorithms for the second and third batch of experiments.

to obtain different outputs and check differences from the original. The third batch only differs in terms of selection of system call to replace, being on the third batch selected randomly instead of iteratively. We perform the same for the two XAI methods separately and do the same for the TN and FP cases. The second and third batch of experiments is presented in Figure 6. The results of the experiments using these algorithms are presented in the section V.

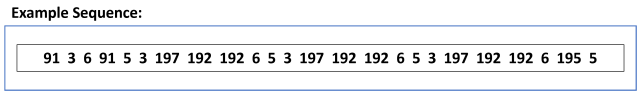


FIGURE 7. Example of sequence of system calls input instance.

V. EXPERIMENTAL WORK

A. DATA PREPROCESSING

The dataset is a generated version of the ADFA-LD dataset. The dataset is based on system calls, with normal traces and attack traces with attacks such as password brute force, denial of service, and network scan. Every instance of the dataset consists of a sequence of 30 system calls. A system call is a programmatic request for a service provided by the operating system, allowing applications to interact with the underlying hardware and access various system resources. Each different system call in the dataset is represented by a number. For instance, the system call ''swapoff'' is represented in the data as ''168''. In the dataset used, there are 149 different system calls. Figure 7 represents an example of an instance of the dataset. It is a sequence of system calls.

One system call might appear multiple times in one sequence. This means that it is possible for one sequence to only have one or two system calls with multiple appearances.

In order to pre-process the system call sequences, two mechanisms are used, the Term Frequency-Inverse Document Frequency (TF-IDF) and the chi-squared test.

One of the most common methods to calculate the weights of words that occur in a document, in document classification, is the TF-IDF model. Term Frequency (TF) refers to the occurrences of a certain word in a document, and Inverse Document Frequency (IDF) pertains to the number of times the word occurs in a document corpus (i.e., collection of documents). The n-gram technique is defined as a tuple of n words where n is, a small positive integer, in our case the n is equal to 2.

For dimension reduction, the chi-squared test is applied. The chi-squared test is often employed to analyze data that is organized into categories or counts. It compares the observed frequencies of each category in a dataset with the frequencies that would be expected if there were no association between the variables. It calculates the statistic called the chi-squared statistic (X2), which measures the difference between the observed and expected frequencies.

Figure 8 shows the pre-processing stage of the input system call sequences. A sequence of system calls is received, suffers a vectorization, followed by a feature reduction, resulting in an array of 150 features, being each feature a constant ngram (label), the value of the features, according to the system call sequence in question. Then, the data is fed to the model.

B. PERFORMANCE EVALUATION

Accuracy is the most common metric used to evaluate classification models. It represents the proportion of correctly classified instances out of the total number of instances in the dataset. It is calculated as:

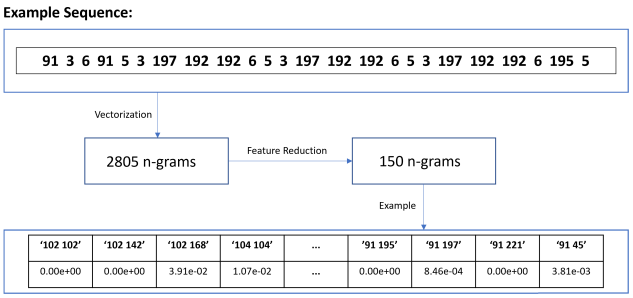


FIGURE 8. System call sequence pre-processing.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{3}$$

Precision is the measure of how many instances predicted as positive are actually TP. In other words, it measures the ability of the model to avoid FP. High precision indicates a low rate of FP, which means the model is more reliable when it predicts a positive result. Precision is calculated as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4}$$

Sensitivity (Recall or TPR) measures the proportion of actual positive instances that are correctly identified as positive. It indicates the ability of the model to detect positive cases. High Sensitivity indicates a low rate of FN, meaning the model is good at identifying positive instances. Sensitivity is calculated as:

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5}$$

Specificity (TNR) measures the proportion of actual negative instances that are correctly identified as negative. It represents the ability of the model to avoid false alarms for negative cases. High Specificity indicates a low rate of FP, indicating that the model is good at correctly identifying negative instances. Specificity is calculated as:

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{6}$$

The F1 Score is the harmonic mean of precision and recall (sensitivity). It provides a balanced measure of a model's performance, taking both FP and FN into account. The F1 Score considers both precision and recall, making it useful when the classes are imbalanced or when both types of errors (FP and FN) are important. It can be calculated as:

$$\text{F}_1\ \text{Score} = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} \tag{7}$$

C. STRUCTURE AND PERFORMANCE OF THE IDS

In order to apply and validate our approach we used as the black box model, a MLPClassifier, part of an IDS solution that performs intrusion detection on IoT devices.

The MLPClassifier has four layers: one input layer and 3 dense layers. The first and second layers of the dense layers

	True Class	
Predicted Class	Positive	Negative
Positive	25,311	475
Negative	2,832	24,038

FIGURE 9. IDS black box model confusion matrix.

have as activation function the "relu" function, while the third dense layer has the "sigmoid" function as activation function. The model is a Tensorflow model and generated with Keras framework. The IDS analyzes sequences of 30 system calls - a fundamental interface between a user-level process and the operating system (OS) kernel. The sequences undergo a pre-processing stage, transforming them in the input data of the MLPClassifier. The model input data is in form of arrays of 150 features, and is explained in more detail further ahead. The model makes binary predictions. If the output of the model, to a certain input instance is 1, then it is because something abnormal has happened, possibly being an intrusion. In the case of the output being 0, no anomaly is detected.

Figure 9, and Table 1, present the current scores of the model. As it can be seen, the model presents good performance values which means that most of its predictions are accurate. However, there are cases where the model is incorrect, as we can see through the confusion matrix. There are 52656 instances on the dataset that we have used. We can see that there are 24 038 TP, which means the number of times that the model predicted that there was a possible anomaly and was correct. The number of FN is 2832, and, as we can understand, very low in comparison to the number of TP. However, the number is relevant and represents important mistakes of the model. In the case of anomaly detection, a FN can be harmful because it is misunderstood by the model and so the cybersecurity experts that are in charge of analyse the system, in case of a potential threat, will not be aware that there is one in these cases. Similarly, the number of TN is 25 311 and the number of FP is 475, which is not as impactful as the TP and FN. This is because a FP was supposed to be a negative prediction, meaning that there was no anomaly happening. In Figure 9 we can see the complete confusion matrix for the model.

As for the scores, the accuracy indicates that the model classifies approximately 93.7% of the instances, which is a good result, especially, taking into account the size of the dataset.

The precision suggests that when the model predicts a positive instance, 98.1% of the time it is correct. The sensitivity value of 89.5% represents that the model correctly identifies that number of percentage of the actual positive instances. A higher value is desirable to minimize FN since it suggests a lower rate of missing positive cases, which is of high importance for the use case of the model. With the value of specificity being 98.2%, we can understand a low rate of FP for negative cases, being the number of FP 475 which is very low in comparison to the size of negative cases. The F1 score with a value of 93.6%, means the model achieves a balanced performance in terms of precision and recall. It indicates a good trade-off between FP and FN. The overall analysis of the model based on the scores is good.

TABLE 1. IDS black box model metrics scores.

Accuracy	Precision	Sensitivity	Specificity	F1 Score
0.9371	0.9806	0.8946	0.9816	0.9356

The model demonstrates appropriate values for the metrics and it seems to be effective in correctly classifying both positive and negative instances. In overall, we can consider that the model presents high score values, which is important for the classification problem it is inserted on.

D. OVERALL RESULTS

In order to address the explainability of the model, we have successfully implemented XAI methods, such as LIME and SHAP. With the implementation of these methods we were able to retrieve explanations about the model results.

With the explanations, we were able to retrieve the most important system calls for the classification of each instance of the dataset. These most important system calls are the ones having the most impact on the classification of a particular instance. On a global analysis, it was possible to observe that there are system calls that only appear, for example, in the explanations of FN cases and not in the explanations of TP cases (and vice versa). This information is useful to cybersecurity experts because one system call that only appears in the explanations of FN cases and not in the explanations of TP cases means that when present in the input, that system call might be misleading the model classification. The same was observed for FP and TN cases. These results are presented in Figure 10. It is important to refer that these results match for both LIME and SHAP. However, on a local scope, for the same instance, the systam calls present in the explanations of LIME might be different than the ones present in the explanations of SHAP. For example, it is possible to see that the system call *setsockopt* appears in the explanations of FN cases and never appears in explanations of TP cases. This might induce that this system call present in the input might be misleading the model to classify the instance as negative when it should be positive.

To evaluate the explanations provided, we have performed perturbations, based on system calls of Figure 10. A system call that only appears, for instance, in the explanations of FN, is replaced every time it appears, with a system call that only appears in the explanations of the opposite case (this being, TP). The same was performed for the TN and FP cases. Some of the most relevant results are represented in Figure 11. For example, when the system call *setsockopt* that is only present in the explanations of FN cases is replaced with the system call *accept* that only appears in the explanations of TP cases, 90% of the instances of FN where *setsockopt* is present in the input change the prediction to TP. This might be a clue

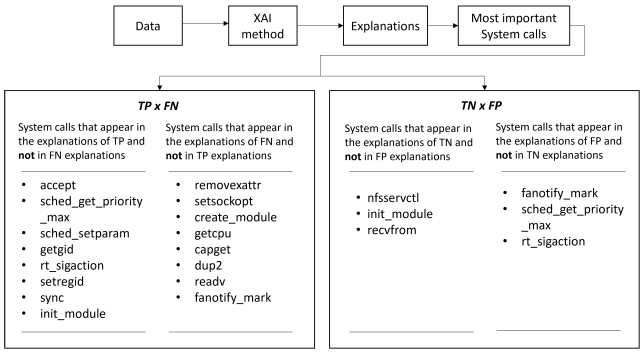


FIGURE 10. System calls that only appear in the explanations of particular cases.

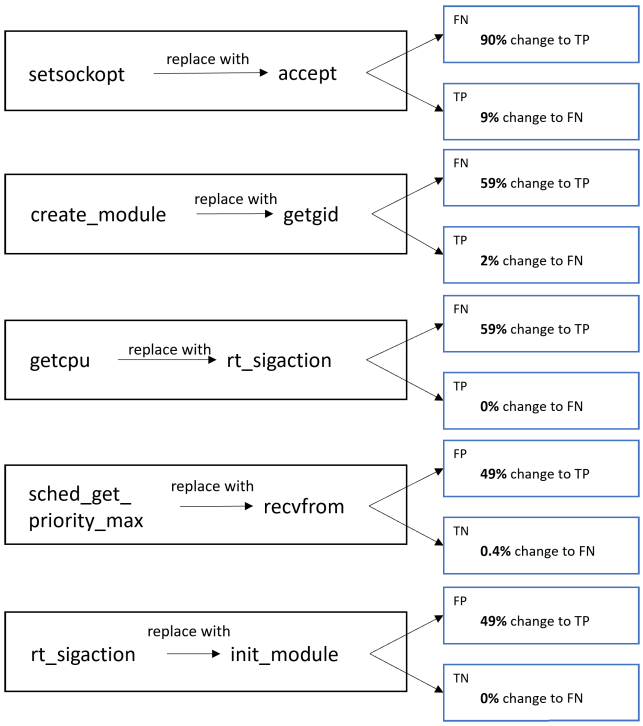


FIGURE 11. Perturbations based on the system calls that only appear in the explanations of opposite cases.

that this system call is having a negative impact on the model classification when present in the input.

With the explanations provided by the XAI methods, the interpretability of the model increases and cybersecurity experts might extract insights about the model behaviour from them. Based on the perturbations carried, the explanations are validated and on an overall analysis, the identified system calls in Figure 10 represent a significant evolution to the interpretation of the model results leading to cybersecurity experts knowing before-hand the impact these features might have when present in the input sequence. Discovering such features that might lead to incorrect predictions, as revealed by LIME and SHAP explanations, assists in refining the data and enhancing the model, thus improving the accuracy of the intrusion detection system. This understanding allows for adjustments to decision thresholds and model improvement, ensuring more effective intrusion detection in real-world scenarios.

TABLE 2. Remove the 10 most important system calls of each instance (LIME).

Cases	Negatives	Positives	Change%
FN	2 517	315	11.12%
TP	8 942	15 096	47.20%
FP	286	189	61.21%
TN	23 798	1 513	5.98%

TABLE 3. Randomly replace the top 10 most important system calls of each instance with system calls that only appear in the explanations of the opposite case (LIME).

Cases	Negatives	Positives	Avg. Change%
FN	1 027	1 805	63.73%
TP	16 613	7 425	69.12%
FP	392	83	82.53%
TN	14 075	11 236	35.11%
Average			**62.62%**

E. LIME RESULTS

For all the experiments, the 10 most important system calls of each instance were retrieved with LIME.

On a first experimental batch, for each instance, the 10 most important system calls were removed from the respective input instances, and the instances were fed back to the black box model to obtain a new result and observe the differences in prediction. The results can be observed in Table 2. In terms of % of changes in prediction, the results are not as satisfactory as expected. However that might be due to the fact of the system call being removed from the sequences.

On a second attempt, the 10 most important system calls are not removed, but replaced. For example, for the FN, the 10 most important system calls were replaced by system calls that only appear in the explanations of TP (opposite of FN). The respective system calls can be seen in Figure 10. The same was performed but on the contrary, and the process was repeated for the FP and TN cases. The results can be viewed on Figure 12. The results represented are just a part of the best and most relevant ones. It is possible to see the impact that the system calls have. For example, in the FP, by replacing the 10 most important system calls for the classification of every instance, with the system call *nfsservctl*, we can observe that about 100% of instance change prediction to TN.

On the final experimental batch, instead of replacing by a fixed system call every time, we have chosen random system calls that only appear in the explanations of the opposite case, to replace. The experiment was performed 5 times due to the randomness of the system calls replacement selection. The results can be observed in Table 3. Besides the TN, the results are satisfactory, and the bad result on the TN might be due to the randomness of the system calls replacing.

On an overall analysis of the LIME results, the impact of different system calls for different scenarios can be observed. These perturbations serve to validate the explanations of LIME as we try to obtain a different output by perturbing

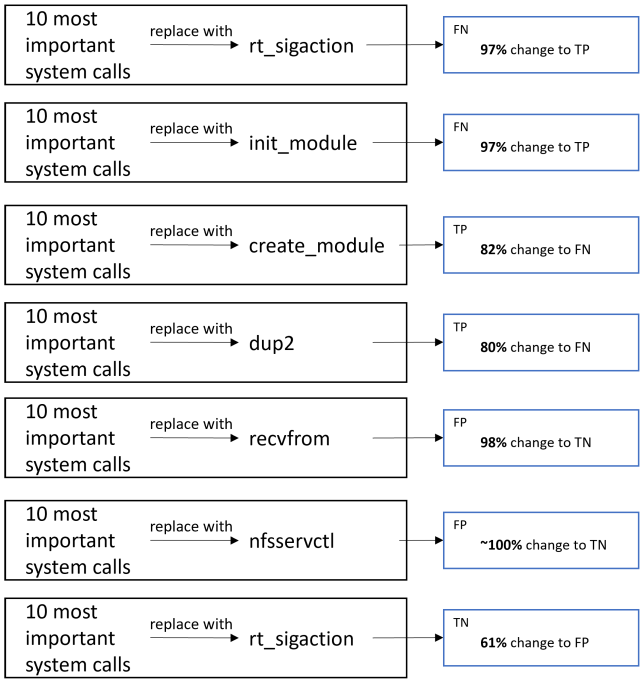


FIGURE 12. System calls that only appear in the explanations of particular cases.

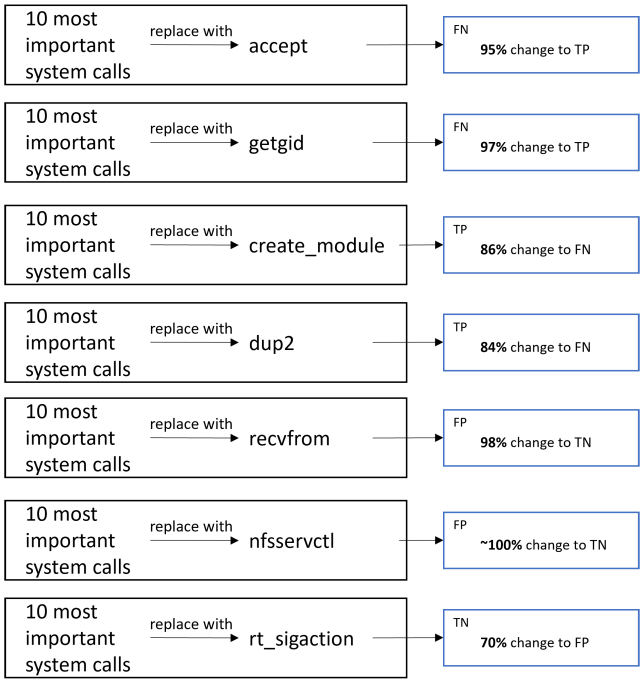


FIGURE 13. System calls that only appear in the explanations of particular cases.

TABLE 4. Remove the 10 most important system calls of each instance (SHAP).

Cases	Negatives	Positives	Change%
FN	2 552	279	9.85%
TP	8 942	15 096	47.20%
FP	286	189	61.21%
TN	23 798	1 513	5.98%

the system calls that LIME says have the most impact for each instance's classification. Not only that but we can also obtain tangible values about the impact of particular system calls that might be affecting the results of the model to a wrong classification when present in the input. It can also be used to compare the precision of the explanations with the ones retrieved with SHAP. It is notable to mention again that this information is significantly important to the cybersecurity experts analyzing the results and helps the continuous improvement of the model.

F. SHAP RESULTS

The experiments with SHAP were similar to the ones with LIME. However, since the methods are different, the results are different as well. For the first batch of experiments (where for all the instances, the 10 most important system calls were removed) the results are presented in Table 4.

For the second batch, the 10 most important system calls of each instance are not removed, but replaced. The experiment is the same as in LIME, but using the explanations of SHAP. The results are presented in Figure 13.

For the final batch, instead of replacing with a fixed system call from the ones that only appear in the explanations of the opposite cases, they are randomly selected from that particular group. The results are presented in Table 5. We can see that by replacing in the input sequence the 10 most important system calls for the classification of each instance with system calls that only appear in the opposite case we are able to obtain significant changes in the output.

TABLE 5. Randomly replace the top 10 most important system calls of each instance with system calls that only appear in the explanations of the opposite case (SHAP).

Cases	Negatives	Positives	Avg. Change%
FN	1 023	1 8051 808	63.86%
TP	16 326	7 711	67.92%
FP	412	63	86.74%
TN	14 075	11 236	44.39%
Average			**65.73%**

In a comparison with the LIME results, it is possible to conclude that the results with the two methods are very similar. There are cases where perturbations considering explanations with LIME have a bigger percentage of changes in prediction. But the same happens in some cases when perturbing based on SHAP explanations. It is important to analyze instances where the two methods produce divergent explanations since it can reveal vulnerabilities in the IDS model, such as susceptibility to adversarial attacks or the presence of non-linear interactions between features. This information can inform strategies to enhance model robustness and resilience against potential threats.

G. COMPARISON OF LIME AND SHAP RESULTS

In order to evaluate, validate, and compare the two XAI methods used (LIME and SHAP), we have performed a survey. The survey had 24 participants. From the participants,

7 had a doctorate degree, 7 a master's degree, and 8 a bachelor's. The majority had worked with machine learning or AI, and were familiar with the concept of XAI. The questions are based on two distinct classification problems. The first is very simple in order to the participants familiarize with LIME and SHAP. The second is the actual IDS classification and explanations examples. The survey can be divided into three main sections. The first section presented the first problem, with examples of inputs and outputs without explanations, with explanations from LIME and SHAP, and additionaly with a validation of the explanations from LIME and SHAP. The second section is similar to the first but using the second presented problem. On the third section, LIME and SHAP were compared directly to observe the differences in terms of interpretability increase and confidence felt by the participants regarding one method over the other. On an overall analysis of the answers, it is possible to conclude that, as expected, having explanations to help in the interpretation of the results of the model is of significant importance.

It is important to note that a choice of method takes into consideration the application context, user preferences, and dataset and model characteristics. In terms of application context, LIME may be more suitable if the priority is local interpretability for individual predictions while SHAP might be a better fit for global interpretability and if consistency is crucial. In terms of user preferences, some users may find LIME's local, user-friendly explanations more accessible, while others might appreciate the theoretical foundations and global interpretability of SHAP. The nature of the dataset and the complexity of the ML model can influence the effectiveness and efficiency of each method, for instance when dealing with large datasets, the scalability of the explanation method becomes important. However, in the survey, in the questions that directly compared the two XAI methods and where the participants had to chose a favorite, LIME is the clear preferred. These questions presented the explanations from LIME and SHAP for the same instance, questioning the participants which they felt more trust and in which they felt the interpretability increased more. When participants were queried about their preferences and specifically sought information about the characteristics of each method, LIME remained the favored choice. This preference is likely attributed to the more user-friendly visual explanations offered by LIME. In the survey, nearly all participants emphasized the significant importance of the visualization of explanations. However we must also highlight that some of the participants who usually voted on SHAP over LIME were the ones with more technical background, elucidating probably to the consistency of the method over the visualizations.

It is notable that generating explanations using LIME and SHAP can be computationally expensive, especially for large-scale IDS deployments with high data throughput. The additional overhead of explanation generation might impact the real-time performance of the system. Comparing both methods in terms of computational load, in our work and experiments, we have to highlight that SHAP was in fact considerably more expensive than LIME, for the exact same experiments. This certainly impacts the choice of the XAI method when one of the factors to look for is the computational load.

We must highlight that the majority of the participants answered that the additional validation step of the explanations increased their trust on the XAI methods and their explanations.

VI. CONCLUSION

In this work, we have used two XAI methods (LIME and SHAP) to provide explanations for the results of a black-box model part of an intrusion detection solution for IoT devices. With the explanations, we were able to identify the most important system calls for the classification of each instance. Based on the explanations provided, we carried out perturbations to the input to observe a different output. With these results, we were able to collect system calls that have a significant negative impact on the output, when present in the input sequence. This way, the cybersecurity person in charge of analyzing the model, is provided with insights about the model behavior, and particular features that when present in the input can lead to wrong predictions.

In this work, we have also performed a direct comparison between the two used XAI methods, LIME, and SHAP. We have performed the same experiments with the two methods and checked which performed better for each experiment. We have also conducted a survey where the participants answered questions that directly compared the two methods in terms of interpretability increase of the black box model, and preference of method, based on characteristics and different provided examples.

The work can be further improved by conducting similar experiments with a broader range of XAI methods, or by implementing the same approach and experiments on different black box models and diverse use cases. Additionally, extending the approach to other types of intrusion detection systems (IDS) beyond IoT, and exploring applications in various domains, could provide valuable insights. Moreover, ongoing validation, evaluation, and comparison of XAI methods and their explanations are essential for advancing the understanding and adoption of transparent AI systems in real-world scenarios.

REFERENCES

[1] L. Santos, C. Rabadao, and R. Gonçalves, "Intrusion detection systems in Internet of Things: A literature review," in *Proc. 13th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Caceres, Spain, Jun. 2018, pp. 1–7, doi: 10.23919/CISTI.2018.8399291.

[2] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018, doi: 10.1016/j.future.2017.07.060.

[3] D. Gunning and D. W. Aha, "DARPA's explainable artificial intelligence program," *AI Mag.*, vol. 40, no. 2, pp. 44–58, 2019, doi: 10.1609/aimag.v40i2.2850.

[4] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73127–73141, 2020, doi: 10.1109/ACCESS.2020.2988359.

[5] S. Patil, V. Varadarajan, S. M. Mazhar, A. Sahibzada, N. Ahmed, O. Sinha, S. Kumar, K. Shaw, and K. Kotecha, "Explainable artificial intelligence for intrusion detection system," *Electronics*, vol. 11, no. 19, p. 3079, 2022, doi: 10.3390/electronics11193079.

[6] D. L. Marino, C. S. Wickramasinghe, and M. Manic, "An adversarial approach for explainable AI in intrusion detection systems," in *Proc. IECON 44th Annu. Conf. IEEE Ind. Electron. Soc.*, 2018, pp. 1–7, doi: 10.48550/arXiv.1811.11705.

[7] S. Mane and D. Rao, "Explaining network intrusion detection system using explainable AI framework," 2021, *arXiv:2103.07110*.

[8] M. Siganos, P. Radoglou-Grammatikis, I. Kotsiuba, E. Markakis, I. Moscholios, S. Goudos, and P. Sarigiannidis, "Explainable AI-based intrusion detection in the Internet of Things," in *Proc. 18th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: Association for Computing Machinery, Aug. 2023, pp. 1–10, doi: 10.1145/3600160.3605162.

[9] P. Barnard, N. Marchetti, and L. A. DaSilva, "Robust network intrusion detection through explainable artificial intelligence (XAI)," *IEEE Netw. Lett.*, vol. 4, no. 3, pp. 167–171, Sep. 2022, doi: 10.1109/LNET.2022.3186589.

[10] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Explainable intrusion detection systems (X-IDS): A survey of current methods, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 112392–112415, 2022.

[11] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach," *Exp. Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 121751.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?' Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 97–101, doi: 10.48550/arXiv.1602.04938.

[13] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017, *arXiv:1705.07874*.

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 1–9, doi: 10.1609/aaai.v32i1.11491.

[15] D. W. Apley and J. Zhu, "Visualizing the effects of predictor variables in black box supervised learning models," *J. Roy. Stat. Soc. Ser. B, Stat. Methodol.*, vol. 82, no. 4, pp. 1059–1086, 2020, doi: 10.48550/arXiv.1612.08468.

[16] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," 2017, *arXiv:1711.00399*.

[17] A. Dhurandhar, P. Y. Chen, R. Luss, C. C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–12, doi: 10.48550/arXiv.1802.07623.

[18] F. K. Došilovic, M. Brcic, and N. Hlupic, "Explainable artificial intelligence: A survey," in *Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, Opatija, Croatia, May 2018, pp. 0210–0215, doi: 10.23919/MIPRO.2018.8400040.

[19] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*, 2nd ed. Independently Published, 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[20] G. Creech, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks," UNSW Sydney, Sydney, NSW, Australia, Tech. Rep., 2014. [Online]. Available: http://hdl.handle.net/1959.4/53218, doi: 10.26190/unsworks/16615.

[21] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 4487–4492.

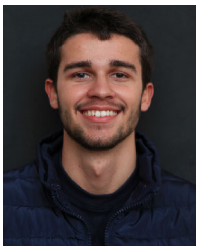
[22] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns," *IEEE Trans. Comput.*, vol. 99, no. 11, 2013.

[23] M. T. Ribeiro. (2016). *Local Interpretable Model-Agnostic Explanations (lime)*. [Online]. Available: https://lime-ml.readthedocs.io/en/latest/index.html

[24] S. Lundberg. (2018). *Welcome To the SHAP Documentation*. [Online]. Available: https://shap.readthedocs.io/en/latest/index.html

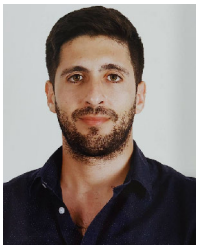
[25] S. Figueiredo, P. Silva, A. Iacovazzi, V. Holubenko, J. Casal, J. M. A. Calero, Q. Wang, P. Colarejo, R. L. Armitt, G. Inches, and S. Raza, "ARCADIAN-IoT–enabling autonomous trust, security and privacy management for IoT," in *Internet of Things* (Lecture Notes in Computer Science), A. González-Vidal, M. Abdelgawad, A. Sabir, E. Ziegler, S. L. Ladid, Eds. Cham, Switzerland: Springer, 2022, pp. 348–359, doi: 10.1007/978-3-031-20936-9_28.

[26] V. Holubenko, P. Silva, and C. Bento, "An intelligent mechanism for monitoring and detecting intrusions in IoT devices," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2023, pp. 959–960, doi: 10.1109/CCNC51644.2023.10060443.



DIOGO GASPAR received the master's degree in informatics engineering and intelligent systems from the University of Coimbra.

He is currently a Researcher and a Software Engineer. His current research interests include machine learning, explainable artificial intelligence, cybersecurity, and cloud computing.



PAULO SILVA received the Ph.D. degree in information science and technology from the University of Coimbra.

He is currently a Researcher and a Software Engineer. He has over 15 publications in journals and international conferences. He has also participated in several European initiatives and projects. His research interests include machine learning, data privacy protection, and security services for cloud computing.

CATARINA SILVA (Senior Member, IEEE) received the Ph.D. degree in computer engineering.

She is currently an Associate Professor with the Department of Informatics Engineering, University of Coimbra, Portugal. She is also a Senior Researcher with the Adaptive Computation Group, CISUC. She has more than 20 years of teaching experience in computer engineering for the B.Sc., M.Sc., and Ph.D. students. More specific recent research projects on Transparency in AI in Finance (COST), Industry 4.0 Circular and Agile Manufacturing and Individualized Consumer Preferences (H2020), Real-Time Condition-Based Maintenance for Adaptive Aircraft Maintenance Planning (H2020), and Real-Time Monitoring of Ambient Air Quality with Low-Cost Nano-Sensors (Sudoe). She is skilled at managing different-sized projects and scientific entrepreneurship, involving people with different backgrounds, namely faculty, students, alumni, and companies. She is the author or coauthor of four books, circa 20 journal articles, and 100 conference papers. Her research interests include machine learning and pattern recognition. She is a scientific committee member and a paper reviewer of several conferences and journals. She is a Board Member of the Portuguese Association of Pattern Recognition, a Senior Member of the IEEE Computational Intelligence Society, and the Past Chair of the IEEE Portuguese Section.