White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/188368/

Version: Accepted Version

Proceedings Paper:
Ioulianou, Philokypros P., Vassilakis, Vassilios G. and Shahandashti, Siamak F. (2022) ML-based Detection of Rank and Blackhole Attacks in RPL Networks. In: 13th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2022:Proceedings. 13th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2022, 20-22 Jul 2022 2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2022. IEEE, PRT, pp. 338-343.

https://doi.org/10.1109/CSNDSP54353.2022.9908049

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

ML-based Detection of Rank and Blackhole Attacks in RPL Networks

Philokypros P. Ioulianou, Vassilios G. Vassilakis, Siamak F. Shahandashti
Department of Computer Science, University of York,
York, United Kingdom

***Abstract*—Although IoT security is a field studied extensively, recent attacks such as BotenaGo show that current security solutions cannot effectively stop the spread of IoT attacks. Machine Learning (ML) techniques are promising in improving protection against such attacks. In this work, three supervised ML algorithms are trained and evaluated for detecting rank and blackhole attacks in RPL-based IoT networks. Extensive simulations of the attacks are implemented to create a dataset and appropriate fields are identified for training the ML model. We use Google AutoML and Microsoft Azure ML platforms to train our model. Our evaluation results show that ML techniques can be effective in detecting rank and blackhole attacks, achieving a precision of 93.3%.**

I. Introduction

In our previous work [1] we proposed a signature-based Intrusion Detection System (IDS) for IoT networks. In this work, the IDS has been enhanced with the addition of a Machine Learning (ML) module that aims at detecting unknown attacks. ML algorithms can be used to train a model for predicting and classifying network traffic. Our ML model is trained using datasets created from network simulations. This is done in order to perform the training using realistic data. Our main goal of creating an ML-based detection module for the IDS is to detect known complex attacks such as a combination of rank and blackhole attacks as well as unknown attacks. In particular, the focus is on networks that rely on the Routing Protocol for Low-Power and Lossy Networks (RPL) [2].

In our previous works [3], [4] we have studied the behaviour and effects of routing and Denial-of-Service (DoS) attacks such as Hello flooding in RPL-based IoT networks. Other important attacks in RPL networks are wormhole and grayhole attacks. They can be detected using methods that rely on packet counts and round-trip times [5] as well as by implementing heartbeat-based protocols [6]. We have also designed and implemented an IDS prototype for protecting RPL networks and devices from battery-drain DoS attacks using rate thresholds [7]. Another recent work has developed a trust-based mechanism that detects and isolates complex routing attacks such as combined rank and blackhole attacks [8]. However, combinations of routing, flooding, and other types of attacks may bypass the threshold-based and trust-based detection and cause serious damage to the network. Therefore, the incorporation of an ML-based module, that has been trained to detect complex or unknown attacks, is expected enhance the detection accuracy.

Many ML algorithms exist for training a model from existing datasets. In this work, Google AutoML [9] and Microsoft Azure ML [10] are used to train the ML model. AutoML is an automated and simple to use tool that provides researchers the opportunity to study and deploy different ML approaches. It has the possibility to scale up an ML model based on the needs of the user. Moreover, it automatically chooses the most appropriate algorithm based on the dataset available. A similar ML-as-a-Service provider is Azure ML. This service allows users to generate and handle custom ML solutions. In addition, it assists users with extending and deploying their workloads to the cloud. An advantage of the Azure ML is that it gives the flexibility to the user to choose a specific ML algorithm from a list of well-known supervised, unsupervised, or semi-supervised algorithms.

The rest of this paper is organised as follows. Section II provides the required background information on the RPL protocol and its related rank and blackhole attacks as well as the basics of the supervised ML algorithms. Section III explains the design of our proposed ML-based detection module. Section IV explains our adopted methodology. In particular, we provide the required details on the dataset creation, pre-processing, packet labelling as well as the specifics of the learning and testing phases. SectionV provides the implementation details. Section VI explains the system configuration and the metrics used for evaluating the ML model. Section VII provides our evaluation results of three ML algorithms. In particular, we calculate the Accuracy, Precision, Recall, F1 Score, and the Area under the Receiver Operating Characteristic Curve (AUC). Finally, Section VIII concludes the paper and discusses possible future directions.

II. Background

A. RPL Protocol

RPL is a standardized routing protocol commonly used in 6LoWPAN [2], [11]. The RPL topology has one central node, the root node, which receives packets from other sensor nodes in the network. RPL is a tree-based proactive routing protocol in which the root node creates a Destination Oriented Directed Acyclic Graph (DODAG) for routing packets [12]. Each node in the DODAG has a unique node ID and a rank which represents the distance of the node from the root. Thus, the rank increases as the packets move downwards from the root node to the peripheral leaf nodes. RPL uses an Objective Function (OF) to derive optimal routes. This could be based on rank, link quality, cost, distance, and other measures.

The DODAG construction begins when the root starts broadcasting DODAG Information Object (DIO) packets to its neighbouring nodes. DIO packets contain relevant information like the OF, sender's rank and IP address, link quality, and other routing metrics [11]. DIO packets are sent periodically to maintain the DODAG stability. Each node broadcasts a DIO packet to neighbouring nodes to construct the upward routes. If any new node wants to join the network, it broadcasts a DODAG

Information Solicitation (DIS) packet. By sending this packet the node is asking if there is any DODAG available, thereby essentially requesting DIO packets from nodes that are already part of the DODAG. RPL can work in the two following modes. In the *non-storing mode*, all the downward routes are stored in a routing table only by the root node. In the *storing mode*, each sensor node maintains and stores the downward routes individually. In our work, the non-storing RPL mode is used for routing packets downwards.

B. Rank and Blackhole Attacks in RPL Networks

An important attack that has been studied in this work is the *rank attack* [13]. In a rank attack a malicious node may intentionally advertise lower rank in order to attract neighbouring devices to select it as preferred parent. A parent node is needed in order to form the DODAG network and allow the creation of routes reaching a Border Router (BR). In cases where networks are small, the best parent is the BR itself. In other cases, metrics such as rank and Expected Transmission Count (ETX) are used to select the best parent. If a malicious node manages to be chosen as the best parent of several nodes, it can attack the network affecting its topology, availability and integrity. As a result, the malicious node can become a single point of failure of the network. In this work, rank attackers advertise fake ranks with a value of 129. This is because the minimum default rank in RPL is 128 and it is assigned to the root. Therefore, nodes will consider a malicious node with a rank of 129 as a good parent for forwarding their packets.

Another important routing attack is the *blackhole attack*, where the malicious node drops all incoming traffic. Therefore, no packets are forwarded from this node and, as a result, network disruption is achieved. By dropping data packets, the retransmission rate of child nodes is increased and an internal DoS attack occurs. This attack could isolate child nodes from the rest of the nodes, thus degrading the performance of the network. In our implementation, all incoming packets are dropped only if they need to be forwarded, and after 2 minutes of simulation are passed.

C. Supervised Machine Learning

In supervised ML the model learns to map input variables X to an output Y using a function $Y = f(X)$. Both input and output datasets are labelled so that the model can learn the relation between the two. During training, input data is provided along with their correct output so that the algorithm learns the patterns. Learning procedure continues until a satisfactory level of performance is reached. After training, the supervised learning algorithm is tested to evaluate its performance. Specifically, unknown data is given as input to the model, and it tries to predict the output value based on the relationships learned from the training procedure.

An example of a classification algorithm that is also used in this work is the *2-class Decision Forest (DF)* [14]. The specific algorithm is a fast supervised ensemble learning model that is often used for predicting two outcomes. The ensemble approach is the one where numerous related models are created and merged in some way together to create a more generalised model instead of depending on a single model only. In this way, better results can be obtained. Creating individual models and combining them together in an ensemble can be achieved in multiple ways. The DF implementation used in this work builds multiple decision trees and merges them together using *voting* to produce more accurate and stable results.

Another classification algorithm that is used in this work is the *2-class Support Vector Machine (SVM)* [15]. SVM is considered as one of the best classification algorithms as it is a reliable algorithm which can achieve high accuracy on predicting the class of unseen data [16]. The SVM algorithm is based on the principle of structural risk minimisation. This principle aims in finding a hypothesis *h* for which one can be sure that the lowest error is observed, whereas other algorithms are using the empirical risk principle which tries to improve the performance of the training dataset.

III. Design

The IDS components designed and implemented in previous works [1], [3], [8] may not be effective in detecting unknown attacks. They have been designed by relying on threshold-based and trust-based mechanisms to detect flooding, rank, and blackhole attacks. For this reason, an ML-based module would be a great feature to add and improve the IDS detection accuracy [17]. Learning from the current behaviour of the nodes will allow the IDS to later identify both known and unknown attacks, achieving higher detection rates. In this work, the focus is to implement an ML-based detection module for detecting a combination of rank and blackhole attacks.

For the generation of training and test datasets, several network simulations have been executed. These are based on two main scenarios, normal and malicious, both using the default Minimum Rank with Hysteresis Objective Function (MRHOF). Specifically, the first scenario is an environment without any attackers. Nodes are operating in a benign fashion and are using MRHOF to choose a parent. In the malicious scenario, one or more rank/blackhole attackers exist. IDS detectors are also deployed in the network to sniff traffic and later help us calculate the metrics. In both scenarios, one BR node, one root node, 30 benign nodes, and a varying number of IDS detectors are deployed. In the malicious scenario, 6 rank/blackhole attackers are deployed in the network.

The packets exchanged in both normal and malicious scenarios have been used during the learning procedure of the ML-based module. Specifically, packet capture (pcap) files generated by the Whitefield framework [18] for each scenario were used. A pcap file contains all network packets exchanged by a specific node. Creating a realistic dataset would need realistic packet captures. Thus, only the pcap files of IDS detectors were collected and analysed during the learning procedure. Any packet sniffed by an IDS detector is recorded into its pcap file.

Through the learning procedure, it was discovered that configuration used for each simulated scenario included specific IDs for each deployed node. For example, if node with ID 3 is an attacker, the ML model will learn that the attacker is always the node with ID 3. In order to avoid this problem and allow the ML model to intelligently detect malicious nodes based on their behaviour and not based on their IDs, a different ID should be assigned to each malicious node in the training and test datasets. If we had the same node ID in both training and test datasets, the ML model would give wrong prediction results. For this

reason, a new simulation configuration was created to assign node IDs randomly to attackers so that nodes have different IDs from the initial configuration. The resulting pcap files from this experiment are used to create the test dataset.

IV. Methodology

The approach followed for building the ML model is described in this section. Each step of our approach is described in the subsections below and a high-level depiction of our adopted methodology is given in Figure 1.

A. Dataset Creation

The first step before training and deploying an ML model, is to produce a dataset for training the model. In order to create a dataset, we have used the network traffic produced during the simulations as described in Section III. The Whitefield framework has the option to export pcap files for each node of a simulation execution. Regarding IDS detectors, they had promiscuous mode enabled in all the experiments so that every packet received in their receiver (RX) range was recorded in the pcap file. A large number of pcap files were generated during each simulation. The next step is to collect only the pcap files generated by IDS detectors for each simulated scenario. This was done because the general concept is to create an ML-based module for IDS using a realistic approach. Achieving that, required us to collect data from devices that actually sniff traffic, and avoid using the simulator's functionalities. Therefore, the ML model is trained and evaluated using the packets received by IDS detectors.

B. Pre-processing

After collecting pcap files from IDS detectors, all files had to be merged into one. For each repetition of each simulated scenario, a pcap file per IDS detector is generated. Therefore, all those files from different simulations were combined together so that a large pcap file per scenario is created.

C. Packet Labelling

Once the dataset is formed, packet labelling is the next step. Each packet is labelled as *malicious* or *benign*. Packet labelling is done based on conditions that are explained in the next sections. Packets contain several fields such as source IP address, destination IP address, protocol, etc. An analysis was carried out on network packets to understand and choose the packet fields that contain useful information. For example, the rank field of DIO packets was selected to identify rank attacks. Packets with rank 129, the rank value of malicious nodes, were labelled as *malicious*. Regarding blackhole attackers, RPL packets with destination a malicious node are labelled also as *malicious*. As for other attacks such as DIS flooding, several simulations indicated that IDS was able to detect them with high detection rate. Therefore, the ML model is designed to focus on complex attacks such as the combination of rank and blackhole attacks. As a last step of this task, processed packets have to be merged together so that one pcap file is created for all the simulated scenarios. The outcome of this process are two files only, the training and test datasets.

The next task is the file conversion from pcap format into Comma-Separated Values (CSV) format. This was necessary as the ML tools allow only CSV files extensions as datasets. Thus, the merged pcap files had to be converted into CSV files. At the end, we have a training and a test datasets in CSV format.

D. Learning and Testing Phases

Training procedure starts right after the datasets are created. Google's Cloud Datastore and Microsoft's Azure Datastore are used to store the training and test datasets in each platform in the cloud. They allow the user to manage, edit, and analyse datasets before proceeding to any training task. Once data pre-processing is finished, the learning procedure starts. AutoML does not indicate which supervised ML algorithms are used to train the ML model. For this reason, the user has to choose only the type of data, which is tabular format in our case, and the target variable for the model, which the algorithm that will try to make predictions. Any other configuration is handled by AutoML and it is automatically adjusted to achieve the highest possible detection performance. The testing procedure begins once the model finishes its training. The test dataset is used during the evaluation process. In particular, the AutoML platform uses the test dataset to evaluate the accuracy of the new model. Moreover, evaluation metrics are generated and analysed, showing the performance of the model on the specified test dataset.

The same process is repeated in the Azure ML platform. The difference, compared to AutoML, is that in Azure ML the user has to design the experiment which includes defining the flow to be followed from feature selection to evaluation model as well as configuring the actual parameters of the deployed ML algorithms. The default parameters provided by the platform have been used in the deployed algorithms of Azure ML. After the evaluation phase, Azure ML allows the user to visualise results, and compare the performance of the deployed algorithms.

V. Implementation

A. Dataset Creation Process

Both training and test datasets are created by executing simulations. In these simulation scenarios, 30 benign nodes, 6 rank/blackhole attackers, and a varying number of 5 to 15 IDS detectors are deployed. Scenarios with 5 and up to 15 detectors are deployed aiming to find an optimal number of detectors. Moreover, in each simulation execution, the deployment of the nodes is random. This means that malicious nodes can be deployed anywhere near benign nodes which makes it a more realistic approach. In particular, 110 simulations are executed (that is, simulations of 11 scenarios are repeated 10 times each) and are analysed for creating the training dataset. Similarly, 110 simulations (11 scenarios repeated 10 times each), are executed and analysed for generating the test dataset. Repeating each scenario 10 times allowed us to collect a large sample for analysis. The seed number plays a significant role in simulations. It affects the behaviour of the nodes in terms of packet processing and packet transmission times. Therefore, we use random seeds in each simulation execution so that random results are produced in each case. We divided the training and test datasets in such as way that the ML model is trained using the former, and tested using the latter. The first 5 repetitions are used for the training dataset, whereas the remaining 5 are used for the test dataset.

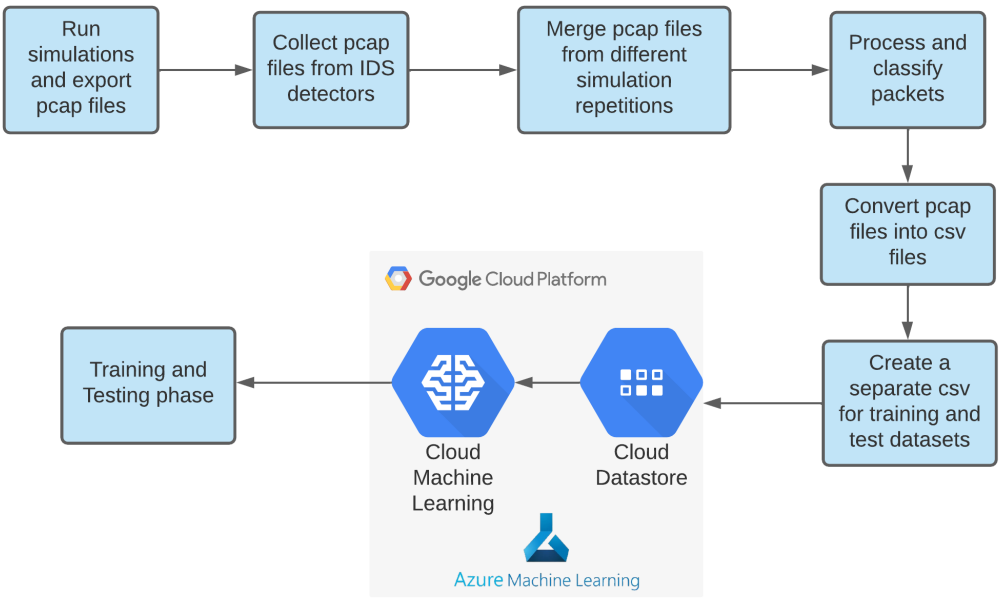

Fig. 1. A high-level description of the methodology followed in this work.

Only pcap files generated by IDS detectors are used to train and evaluate the ML model. This is to ensure that the ML model is as realistic as possible by using the packets captured from the IDS detectors during the simulations.

After the dataset analysis, the number of benign packets is higher than the number of malicious packets. This is expected because the number of attackers deployed in the network is less than benign nodes. Therefore, there is an imbalance in the created datasets which the selected classification algorithms in the training phase will need to handle.

In order to create both training and test datasets, a script was written in Python language so that pcap files are parsed and analysed. A prerequisite for generating the datasets is to export and parse pcap files created by the simulations explained in previous section.

Algorithm 1 presents the steps involved in order to create the training and test datasets. As both datasets follow the same procedure, the following describes the creation of the training dataset only. Assuming pcap files are available, the first step, as shown in Algorithm 1, is to merge all pcap files of the training dataset into one file by calling the *merge_training_data_into_pcap()* function. Inside this function, the script iterates over each scenario to combine and merge pcap files from the first 5 repetitions created by the simulated scenario. As mentioned earlier, only pcap files generated by IDS detectors were used during the dataset creation procedure.

The output of this function is a pcap file per scenario that contains all packets captured from IDS detectors. The next step is to call the *create_training_csv()* for each scenario to rename packet fields, classify packets, filter packet types, and finally create a CSV file. Specifically, this function renames packet headers for better understanding. Then, it classifies packets into *malicious* or *benign*. This is significant as it will allow the ML algorithm to learn when a packet belongs to each category.

Algorithm 1 Dataset creation process

```
1: Input: Pcap files exported from simulations
2: Output: Training and test datasets as CSV files
3: merge_training_data_into_pcap();
4: merge_test_data_into_pcap();
5: for each sim in SCENARIOS do
6:     create_training_csv(sim);
7:     create_test_csv(sim);
8: end for
9: for each file in SCENARIOS_CSV do
10:    merge_and_filter_training_csv(file);
11:    merge_and_filter_test_csv(file);
12: end for
```

We created a new field in the CSV file called *Category*. This field indicates that a packet is malicious if any of the following conditions occurs:

- The rank field has a value equal to 129.
- The next-hop destination IP address belongs to a rank/blackhole attacker.

The first condition means that a rank attacker advertises false rank while second condition means that the next-hop node is an attacker. Packets that do not fulfil any of the above conditions are treated as benign. The last task of the function is to filter packets so that only ICMPv6 and UDP packets are contained in the CSV file. This aims to remove any irrelevant packets captured by nodes. The resulting file is a CSV file that contains many useful information such as source and destination IP addresses, protocol, timestamp, and other fields. As each scenario generated a different CSV file, we had to merge all CSV files into one large file. This process is done by *merge_and_filter_training_csv()* function. The output of this function is a CSV file with many fields available for training

TABLE I
PACKET FIELDS IN TRAINING AND TEST DATASETS

Field name	Description
Rank	DODAG rank value of the node sending the DIO packet
wpanDst	MAC layer destination address in hexademical
wpanDst16Int	MAC layer destination address in decimal
Src	Source IP address of the sending node
Dst	Destination IP address of the sending node
Parent	Parent IP of the sending node
Code	RPL message type
Flag	Packet flags
wsInfo	Additional information about the packet
ipv6Plen	IPv6 packet payload length
Length	Frame length
ipv6Nxt	IPv6 packet next header
Time	Absolute time when this frame was captured
DAOSequence	A sequence number incremented at each unique DAO message from a node and echoed in the DAO-ACK packet
Reserved	Reserved flag, must be zero
rplInstance	Shows which RPL Instance the DODAG is part of

the ML model. As indicated in Algorithm 1, the procedure described for creating the training dataset is also repeated for creating the testing dataset.

B. Packet Processing

Each network packet sent by a node contains a large number of fields from PHY, MAC, 6LoWPAN, network, transport, and application layers. However, for our experiments a total of 16 fields were chosen to be included in the datasets. The packet fields contained in both training and test datasets are shown in Table I. It can be observed that some fields contain basic network information, whereas other fields contain RPL-specific information. The selection of these fields was based on detailed packet analysis. Several pcap files were analysed to decide which fields contain critical information. Apart from that, the way of how routing attacks work was considered to create the list of 16 packet fields. For example, the *Rank* and *ipv6Nxt* fields are useful to attackers as they are often modified during routing attacks to advertise fake ranks or to route packets to blackhole nodes.

VI. CONFIGURATIONS AND METRICS

The configurations and metrics used for evaluating ML models are discussed in this section. Transformation of data was performed in both platforms to normalise data and make them appropriate for the ML algorithms. Moreover, the threshold for the learning rate was configured at 0.5 in both platforms.

As the Azure ML is a platform that implements several well-known supervised algorithms, the user has to choose a specific ML algorithm to train and test an ML model. As the current task was to classify packets into two categories, the decision for choosing the training algorithms was based on accuracy and training time [19]. The first chosen algorithm is *2-class DF* which usually shows high accuracy but needs moderate training time [14]. The second algorithm is *2-class SVM* [15] which is generally considered a good choice for training models with large feature sets but usually shows less accuracy than DF. Regarding the AutoML, the user does not select any algorithm as Google uses its own proprietary ML algorithms. Therefore, the only available configuration for the user is to choose the structure of the data and set the input datasets.

Both 2-class SVM and DF algorithms are configured and executed on the Azure ML platform. The default parameters provided by the platform have been used. Specifically, in the SVM algorithm the number of iterations for building the model was chosen to be 1, the lambda value which is used to tune the model was set to 0.001, and the features are chosen to be normalised before training. Regarding the DF algorithm, the main parameter that user has to select is the *resampling* method. This is the method that the algorithm is using to generate the trees. Bagging or bootstrap aggregating is the method we selected for our experiments. In this method, each tree is developed on a new sample, which is formed by randomly sampling the original dataset with replacement until the dataset becomes the same size as the original. The outputs of the trees are aggregated using the *voting* method. This means that trees with high predictions will be given more weight in the final decision. As for the remaining configuration, the maximum number of decision trees was set to 8, the maximum depth of any decision tree to 32, and the minimum number of samples per leaf node to 1. The two algorithms used the system clock value as a random seed.

The evaluation has been performed among the AutoML, 2-class SVM and 2-class DF models. In regards to the metrics, the following were used to evaluate ML models:

- **AUC**: It is the Area under the Receiver Operating Characteristic (ROC) Curve. AUC value ranges from zero to one and the higher it is, the better.
- **Accuracy:** It is the ratio of the number of true predictions to the total number of cases. It is defined as the probability that the IDS outputs correctly when the behaviour of the system is normal and malicious.
- **Precision:** It is the ability of a model to avoid labelling negative samples as positive. It is defined as the probability that there is an attack when the IDS outputs an alert. In other words, it is the ratio of correctly predicted positive observations to the total predicted positive observations. Low precision indicates high False Positive (FP) rate.
- **Recall:** It is the ratio of correctly classified as positive samples over by the total number of positive samples. It is defined as the probability that the IDS outputs an alert when there is an attack. In other words, Recall is the True Positive (TP) rate.
- **F1 score:** It tells how precise and robust the classifier is. In particular, F1 score is the harmonic mean of precision and recall metrics. It provides a fair representation of both false positives and false negatives. However, true negatives are not taken into account in the calculations.

VII. EVALUATION RESULTS

Results obtained from the ML algorithms are presented in this section. Table II depicts the three ML algorithms used in the experiments along with the calculated evaluation metrics. As mentioned in previous sections, the test dataset is used as an input data to evaluate the trained models. We observe that the AutoML algorithm shows superior performance in comparison with SVM and DF. Evaluation showed a Precision of 93.3% in AutoML, followed by 76.7% and 3.5% in DF and SVM, respectively. AutoML does not provide the Accuracy metric, SVM achieves 76.1%, whereas DF achieves 92.2%. The F1

TABLE II
ML ALGORITHMS EVALUATION RESULTS

Metric	2-class SVM MS Azure ML	2-class DF MS Azure ML	Google AutoML
Accuracy	76.1%	92.2%	-
Precision	3.5%	76.7%	93.3%
Recall	2.8%	62%	93.3%
F1 Score	3.1%	68.6%	93.3%
AUC	0.49	0.84	0.92

Score is one of the most important metrics to look for when classification algorithms are evaluated because is the harmonic mean of Recall and Precision metrics. According to Table II, the highest F1 Score is 93.3% achieved by AutoML, followed by 68.6% of DF, and the lowest one is 3.1% of SVM. Another metric is AUC which shows if the model can discriminate between malicious and benign packets. Based on the results, AutoML has 0.92 while 0.84 and 0.49 are recorded for DF and SVM algorithms, respectively.

Another interesting metric generated by AutoML is the feature importance. AutoML analysed the dataset to find the most important features after the training phase of the ML model. Based on the results, only 3 out of 16 features are important to consider. Specifically, the *Rank* feature is more than 80% important, the *wpanDst* field is less than 20% important, and the *Dst* has very low importance. The *Rank* field is important to be included as the rank attack modifies this specific field. For the other two fields, further analysis should be made to determine if they are really useful for training the ML model. Although the remaining fields seem to have low importance, some might be needed to avoid data overfitting. Therefore, careful analysis of the features should be done before proceeding to feature selection which will enhance the model's performance.

Generally speaking, the low performance of SVM and DF algorithms could be due to the imbalanced datasets created for training purposes. The default settings of ML algorithms trained the model in a way that low performance is achieved as depicted by the metrics. On the other hand, AutoML produced better results and handled the imbalanced dataset in a way that it did not affect its performance. For this reason, the ML model created by AutoML is the preferable candidate to be deployed as part of the IDS detection module.

VIII. CONCLUSION

We examined the effectiveness of publicly available ML frameworks in detecting a combination of rank and blackhole attacks in RPL networks. Specifically, Google AutoML, 2-class SVM, and 2-class DF algorithms were used to train and evaluate the ML model. Simulations were implemented and a dataset was created as the ground truth to train the models. Evaluation results showed that the models trained using the AutoML framework achieved a Precision of 93.3%. Such ML-based detection modules can be integrated in IDS platforms to augment existing detection capabilities based on other techniques. Further optimisations and designing dedicated ML algorithms for the examined as well as other attacks in RPL networks are promising directions of research in this area.

REFERENCES

[1] P. P. Ioulianou, V. G. Vassilakis, I. D. Moscholios, and M. D. Logothetis, "A signature-based intrusion detection system for the Internet of things," in *IEICE Information and Communication Technology Forum (ICTF)*, Graz, Austria, July 2018, pp. 1–6.

[2] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. K. Alexander *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks." *RFC*, vol. 6550, pp. 1–157, March 2012.

[3] P. P. Ioulianou and V. G. Vassilakis, "Denial-of-service attacks and countermeasures in the RPL-based Internet of Things," *2nd International Workshop on Attacks and Defenses for Internet-of-Things (ADIoT) in conjunction with ESORICS*, Luxemburg, Sept. 2019.

[4] P. P. Ioulianou, V. G. Vassilakis, and M. D. Logothetis, "Battery drain denial-of-service attacks and defenses in the Internet of things," *Journal of Telecommunications and Information Technology*, vol. 2, pp. 37–45, April 2019.

[5] C. Samuel, B. M. Alvarez, E. G. Ribera, P. P. Ioulianou, and V. G. Vassilakis, "Performance evaluation of a wormhole detection method using round-trip times and hop counts in RPL-based 6LoWPAN networks," in *12th IEEE/IET International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Porto, Portugal, July 2020, pp. 1–6.

[6] E. G. Ribera, B. M. Alvarez, C. Samuel, P. P. Ioulianou, and V. G. Vassilakis, "Heartbeat-based detection of blackhole and greyhole attacks in RPL networks," in *12th IEEE/IET International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Porto, Portugal, July 2020, pp. 1–6.

[7] R. Smith, D. Palin, P. P. Ioulianou, V. G. Vassilakis, and S. F. Shahandashti, "Battery draining attacks against edge computing nodes in IoT networks," *Cyber-Physical Systems*, vol. 6, no. 2, pp. 96–116, January 2020.

[8] P. P. Ioulianou, V. G. Vassilakis, and S. F. Shahandashti, "A trust-based intrusion detection system for RPL networks: Detecting a combination of rank and blackhole attacks," *Journal of Cybersecurity and Privacy*, vol. 2, no. 1, pp. 124–153, March 2022.

[9] "Google Cloud AutoML Custom Machine Learning Models," https://cloud.google.com/automl/, accessed: 30 May 2022.

[10] "Microsoft Azure Machine Learning," https://azure.microsoft.com/en-us/services/machine-learning/, accessed: 30 May 2022.

[11] D. Airehrour, J. Gutierrez, and S. K. Ray, "Secure routing for internet of things: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 198–213, May 2016.

[12] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-based internet of things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, p. 11, August 2013.

[13] M. A. Boudouaia, A. Ali-Pacha, A. Abouaissa, and P. Lorenz, "Security against rank attack in RPL protocol," *IEEE Network*, vol. 34, no. 4, pp. 133–139, July/August 2020.

[14] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, A. Criminisi, A. Criminisi, D. Zikic, and J. Shotton, "Decision forests, convolutional networks and the models in-between," in *arXiv:1603.01250*, March 2016.

[15] S. Suthaharan, "Support vector machine," in *Machine Learning Models and Algorithms for Big Data Classification. Integrated Series in Information Systems*, 2016, vol. 36, pp. 207–235.

[16] Q. Yang and F. Li, "Support vector machine for intrusion detection based on LSI feature selection," in *6th IEEE World Congress on Intelligent Control and Automation*, Dalian, China, June 2006, pp. 4113–4117.

[17] A. M. Pasikhani, J. A. Clark, P. Gope, and A. Alshahrani, "Intrusion detection systems in RPL-based 6LoWPAN: A systematic literature review," *IEEE Sensors Journal*, vol. 21, no. 11, pp. 12 940–12 968, March 2021.

[18] R. Jadhav, "Whitefield framework," https://github.com/whitefield-framework/whitefield, accessed: 30 May 2022.

[19] Microsoft, "How to select algorithms for Azure Machine Learning," https://docs.microsoft.com/en-us/azure/machine-learning/how-to-select-algorithms, January 2022, accessed: 30 May 2022.