

Malware families discovery via Open-Set Recognition on Android manifest permissions

Filippo Leveni^{✉,a}, Matteo Mistura^c, Francesco Iubatti, Carmine Giangregorio^c, Nicolò Pastore^c,
Cesare Alippi^{a,b} and Giacomo Boracchi^{b,a}

^a*Department of Electronics, Information and Bioengineering. Politecnico di Milano - Milan, Italy*

^b*Faculty of Informatics. Università della Svizzera Italiana - Lugano, Switzerland*

^c*Cleafy S.p.A. - Milan, Italy*

Abstract. Malware are malicious programs that are grouped into families based on their penetration technique, source code, and other characteristics. Classifying malware programs into their respective families is essential for building effective defenses against cyber threats. Machine learning models have a huge potential in malware detection on mobile devices, as malware families can be recognized by classifying permission data extracted from Android manifest files. Still, the malware classification task is challenging due to the high-dimensional nature of permission data and the limited availability of training samples. In particular, the steady emergence of new malware families makes it impossible to acquire a comprehensive training set covering all the malware classes. In this work, we present a malware classification system that, on top of classifying known malware, detects new ones. In particular, we combine an open-set recognition technique developed within the computer vision community, namely MAXLOGIT, with a tree-based GRADIENT BOOSTING classifier, which is particularly effective in classifying high-dimensional data. Our solution turns out to be very practical, as it can be seamlessly employed in a standard classification workflow, and efficient, as it adds minimal computational overhead. Experiments on public and proprietary datasets demonstrate the potential of our solution, which has been deployed in a business environment.

1 Introduction

Malware programs are designed to disrupt, damage, or gain unauthorized access to computer systems or data. Android, the most used mobile device operating system [21, 22], has gained the attention of malware developers thanks to its popularity and increased use for business and financial activities. Malware programs have very different behaviors and characteristics, thus are classified into distinct families based on their attributes. Classifying malware is very important to enable quick identification and an effective response to potential attacks. Furthermore, analyzing malware families provides insights into evolving attack trends and patterns, informing security researchers with an updated view of the threat landscape, thus helping organizations assess their risk exposure, prioritize security efforts, and allocate resources more effectively to protect themselves and their customers.

Malware programs targeting Android devices often exploit permissions declared in the manifest file to gain unauthorized access to sensitive resources [18]. These malicious applications may request

permissions beyond what is necessary for their stated functionality, tricking users into granting access to sensitive data or system resources. For example, malware programs can leverage permissions to carry out activities such as sending SMS messages, stealing banking or payment app data, or downloading and executing additional malicious payloads. By abusing permissions declared in the manifest file, malware programs can operate stealthily, posing a significant threat to users' privacy and security.

Malware classification often relies on rule-based approaches crafted by experts [18], but these are ineffective in identifying complex patterns in the huge space of permission requests and are prone to introducing human biases. Conversely, machine learning models learn distinctive patterns directly from data, leading to a more accurate malware classification. Permission information is typically one-hot encoded, resulting in a very sparse and high-dimensional binary feature vector for each malware program. In such high-dimensional settings, tree-based classifiers like decision trees and gradient boosting machines are considered state-of-the-art, as they are particularly effective and efficient in capturing complex nonlinear relationships between binary features [13, 23].

Malware developers continuously craft new malicious programs to elude classification by security systems. The emergence of new malware families poses significant challenges to cybersecurity, as these cannot be classified from a previously trained model. Therefore, a robust malware classification system must effectively identify known classes and also detect unknown ones, to ensure overall protection for Android devices. This challenge, known as Open-Set Recognition (OSR), is typically addressed by assessing the classifier's degree of uncertainty. OSR has been widely addressed within the computer vision community, thus most solutions are tailored for neural network classifiers [2, 7, 15], which are not well suited for malware classification. To the best of our knowledge, OSR methods have not been extended to tree-based classifiers that are widely used in malware classification.

In this work, we present a practical and efficient solution for identifying unknown malware families – namely those not represented in the training set – by extending MAXLOGIT, a simple yet widely used OSR technique developed for neural networks, to tree-based GRADIENT BOOSTING classification models. In contrast with recent malware OSR solutions, which involve sophisticated deep learning architectures often requiring ad-hoc training procedures, such

as Generative Adversarial Networks (GANs) [8], Transformer-based models [16], or multimodal deep embeddings [9], our solution can be seamlessly integrated into an existing tree-based malware classifier, without even modifying the training procedure. The above-cited methods poorly fit with real-world industrial scenarios as they often require large volumes of training data, high computational costs, complex training procedures, and lack of compatibility with existing systems.

Specifically, we show that MAXLOGIT can be readily applied on decision values already produced by a GRADIENT BOOSTING classifier, enabling OSR on tree-based models – widely adopted in malware detection systems – without requiring any changes to the existing pipelines. Our solution is, therefore, tailored for resource-constrained industrial environments, prioritizing low latency, data efficiency, and ease of integration with existing infrastructures. We validate our solution on both public and proprietary real-world datasets, demonstrating its effectiveness and superior performance over a nearest-neighbor OSR baseline [17] in handling high-dimensional binary data, which can be a viable alternative to ours, as it can also be seamlessly integrated into a pre-trained tree-based classifier. Importantly, our solution has been successfully deployed in a business environment, and it is currently part of their engine.

2 Background

Malware classification approaches typically rely on heuristic rules, and can be divided in static, such as signature-based and permission-based, and dynamic [18]. Dynamic analysis involves observing application behavior within a sandbox environment, and monitoring system calls to construct a function call graph which is analyzed to identify malicious behaviour. Some dynamic approaches employ machine learning algorithms to classify malware programs, usually trained using function call graphs as input [10]. Depending on the classification model, this approach can easily integrate with open-set recognition techniques to discover new malware families [11, 14]. Despite the high potential of dynamic analysis, the runtime testing required to construct the function call graph makes it inefficient and unsuitable for high throughput scenarios.

In signature-based approaches, unique identifiers of known malware programs are stored in a database, and any application is compared to them and eventually flagged as malware [6]. Although signature-based analysis proves efficient, it is effective only for malware families that are already stored in the database, making it not appropriate to identify new ones.

Permission analysis, on the other hand, consists in classifying programs based solely on the permissions they request from the operating system [19]. In this work, we focus on permission analysis as it is more flexible than signature-based analysis and more efficient compared to dynamic analysis.

Android applications specify required permissions in a mandatory file named `AndroidManifest`, which includes both custom and system permissions. Custom permissions do not require access to sensitive data such as contacts or filesystem, whereas system permissions encompass all permissions exposed by the system, with only the most sensitive ones requiring explicit user approval. Our study focuses solely on the latter, due to their potential security risks for the user.

The permission extraction process involves filtering out custom permissions and applying one-hot encoding to system permissions, enabling malware analysis through a machine learning model. One-hot encoding is a common preprocessing method, where each per-

mission is represented by a binary value (1 for requested, 0 for not requested). This results in each application being represented by a high-dimensional binary vector of length P , where P is the total number of permissions. However, the limited number of samples typically available in real-world malware classification scenarios (small n), coupled with the high-dimensional feature space (large P) poses challenges for effectively training classification models.

3 Problem formulation

A malicious application is represented by a vector of permissions $\mathbf{p} \in \{0, 1\}^P$, where P is the total number of permissions considered, and $p_i = 1$ if the i -th permission is listed in the application manifest. These malicious applications might either belong to a known class $\ell \in \mathcal{L}$ indicating a known malware family, or to a novel family that has never been observed before.

Our goal is to train an open-set classifier \mathcal{K} that associates to each malicious application \mathbf{p} either a known class label $\hat{\ell}(\mathbf{p}) \in \mathcal{L}$ or the *Novel* label, *i.e.*:

$$\mathcal{K}(\mathbf{p}) = \begin{cases} Novel \\ \hat{\ell}(\mathbf{p}) \in \mathcal{L}. \end{cases} \quad (1)$$

We assume that we are provided with a training set $\mathcal{TR} = \{(\mathbf{p}_i, \ell_i) \mid \ell_i \in \mathcal{L}\}_{i=1, \dots, n}$ of annotated malicious applications belonging to known families and with a test set $\mathcal{TS} = \{(\mathbf{p}_i, \ell_i) \mid \ell_i \in \mathcal{L} \cup \{Novel\}\}_{i=1, \dots, m}$ of annotated malicious applications belonging to both known and novel families.

4 Related work

Using the manifest file permissions alongside machine learning models for malware classification is a well-established practice. In [19, 13], the permissions vector serves as behavioral marker and it is fed to machine learning models ranging from Support Vector Machines and Gaussian Naive Bayes to Random Forests. Their findings indicate that machine learning models trained solely on manifest file permissions significantly outperform traditional anti-virus engines, with Random Forest achieving the highest accuracy. In [23], they employ a tree-based GRADIENT BOOSTING model to classify six malware classes using three permission categories, highlighting that boosting is particularly suited for malware classification. However, all these methods operate as closed-set classifiers and are therefore unable to discover new malware families.

An effective machine learning malware classification system should both accurately classify known malware programs and identify novel, unknown malware families. This challenge, known as OSR [2], is typically addressed by assessing the classifier’s degree of uncertainty. OSR systems can be categorized into two types. The first type distinguishes between instances of known families and unknown ones, but does not differentiate among known families [20, 3, 4]. This kind of OSR approach, also referred to as *anomaly detection*, does not address the known malware classification task, which is a primary requirement in our setting. Conversely, the second type of OSR systems can both classify known families as well as identify instances of unknown ones [2, 5].

The OSR problem has been extensively studied in the computer vision community [2, 7, 15, 24], with most solutions relying on convolutional neural networks. Neural networks have also been used for OSR in malware detection [11, 14], where applications are executed in a sandbox to extract function call graphs, which are then converted into adjacency matrices. These matrices are used to train a

convolutional neural network using various loss functions to learn a discriminative representation of malware families in a latent space. During testing, an instance is classified as unknown if the distance to its closest centroid exceeds a threshold. While effective, the extraction of function call graphs is time-consuming, making it impractical for high throughput scenarios. Alternative approaches to extend closed-set classification systems for OSR have also been proposed. In [25] they use permission-based fingerprinting to classify samples of known Android malware families and then apply heuristics-based filtering to identify specific behaviors exhibited by unknown malicious families. When an application classified as malicious does not appear in the database, they consider it as novel and generate the corresponding permission-based footprint in a feedback loop. However, their heuristic-based approach targets only specific Android features that may be exploited to load new code, thus limiting the ability to identify different types of unknown malware programs.

Recent advances in OSR for malware detection have moved toward increasingly complex deep learning architectures. For instance, in [8] they propose a Conservative Novelty Synthesizing Network based on GANs to generate marginal malware samples that help distinguish unknown families. In [16] they introduce DOMR, a Transformer-based OSR method that relies on episodic training and meta-learning, while in [9] they further extend OSR to multimodal settings using dual-embedding networks combining CNNs and BERT-like Transformers. Although these methods report strong performance on large-scale datasets, they rely heavily on abundant labeled data, high-end computing resources, and ad-hoc training procedures, making them unsuitable for many industrial environments.

In industrial environments, it is preferable to address OSR using models trained through efficient procedures that require minimal changes to existing closed-set classifiers, facilitating their deployment and maintenance. A lightweight example is Open-Set Nearest-Neighbor (OSNN) [17], which extends a 1-NN classifier to address the OSR task. OSNN computes the ratio of distances between a sample and its two nearest neighbors from different families and classifies the sample as unknown if the ratio falls below a specified threshold. However, our experiments show that OSNN performs poorly on high-dimensional data. Tree-based models like GRADIENT BOOSTING, on the other hand, are widely used in cybersecurity due to their efficiency and strong performance on high-dimensional sparse data like one-hot encoded Android permissions, integrating well with existing infrastructures and industrial pipelines.

In this work, we combine the effectiveness and efficiency of GRADIENT BOOSTING in classifying malware programs, with the MAXLOGIT OSR technique, originally developed within the computer vision community. Our approach classifies malware instances into known families while detecting unknown ones by relying exclusively on permission analysis, making it faster than methods that require function call graph extraction and manipulation. To the best of our knowledge, OSR techniques have not been applied to tree-based classifiers, and by extending MAXLOGIT to operate with GRADIENT BOOSTING, our work fills this gap providing an OSR solution that is practically feasible in real-world settings.

5 Proposed approach

Our proposed open-set classifier \mathcal{K} , depicted in Figure 1, consists of a closed-set classifier \mathcal{C} and an open-set recognition module \mathcal{O} . The closed-set classifier assigns to each malicious application \mathbf{p} a known class label, expressed as $\mathcal{C}(\mathbf{p}) = \hat{\ell}(\mathbf{p}) \in \mathcal{L}$. The open-set recognition module $\mathcal{O}(\mathbf{z}_p) \in \{Novel, Not\ novel\}$ determines whether \mathbf{p}

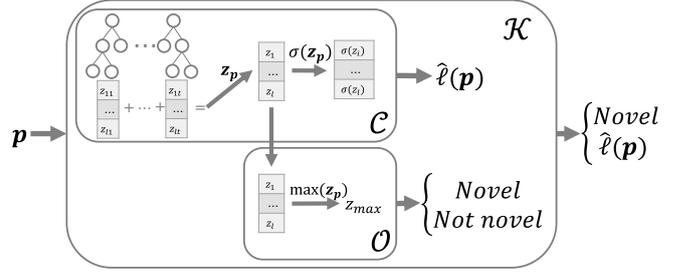


Figure 1: Depiction of our proposed open-set classifier \mathcal{K} .

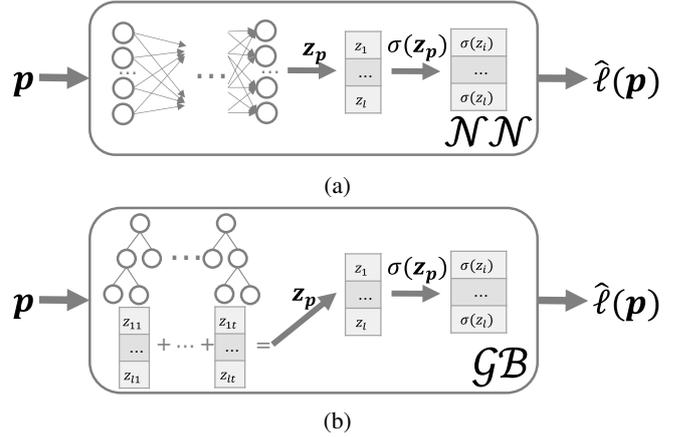


Figure 2: Analogy between the LOGITS vector in a neural network-based classifier (a) and the decision values in a tree-based GRADIENT BOOSTING classifier (b). In both cases, the vector \mathbf{z}_p represents the raw output values before applying the softmax function $\sigma(\mathbf{z}_p)$.

belongs to a *novel* (unknown) or *not novel* (known) class based on the raw output values \mathbf{z}_p assigned to \mathbf{p} by the classifier \mathcal{C} . Therefore, we can restate (1) in the following way:

$$\mathcal{K}(\mathbf{p}) = \begin{cases} Novel & \text{if } \mathcal{O}(\mathbf{z}_p) = Novel \\ \hat{\ell}(\mathbf{p}) & \text{otherwise.} \end{cases}$$

In our approach, we employ tree-based GRADIENT BOOSTING [12] as closed-set classifier, that is a set of boosted classification trees $\mathcal{C} = \{T_i\}_{i=1, \dots, t}$, and MAXLOGIT [24] as open-set recognition module \mathcal{O} , that is a threshold $\tau \in \mathbb{R}$ set on the classifier’s raw output values \mathbf{z}_p to ensure a specified false alarm rate.

5.1 LOGIT extraction

MAXLOGIT has been originally developed within the computer vision community and has traditionally been coupled with neural network-based classifiers. In this context, MAXLOGIT operates on the raw output scores $\mathbf{z}_p = [z_1, \dots, z_l]$ generated by the last layer of the neural network, known as the LOGITS, before the softmax function. In Figure 2a we illustrated a neural network classifier, where \mathbf{p} is the input point and $\hat{\ell}(\mathbf{p})$ is the predicted label. Since the softmax does not change the order of the scores in \mathbf{z}_p , the predicted class $\hat{\ell}(\mathbf{p})$ corresponds to the entry having the maximum value within the probability vector $\sigma(\mathbf{z}_p) = [\sigma(z_1), \dots, \sigma(z_l)]$, which denotes the output of the softmax function to each element z_i of the LOGITS vector \mathbf{z}_p , i.e.:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^l e^{z_j}}. \quad (2)$$

MAXLOGIT approach operates directly on the raw output values \mathbf{z}_p because LOGITS display clearer separation between known and unknown classes when compared to normalized probability values $\sigma(\mathbf{z}_p)$ [24]. Unfortunately, neural networks are not well suited for handling high-dimensional sparse binary data. Hence, we opted for tree-based GRADIENT BOOSTING as the classification algorithm due to its effectiveness in managing such data. To the best of our knowledge, this is the first time that the MAXLOGIT approach is used in conjunction with classifiers other than those based on neural networks.

GRADIENT BOOSTING classifier does not have proper LOGITS, but it produces decision values, which are essentially the raw, unnormalized scores assigned to each class by the ensemble model before being transformed into probabilities (Figure 2b). In GRADIENT BOOSTING, in order to obtain the final prediction $\hat{\ell}(\mathbf{p})$ for a given input sample \mathbf{p} , decision values are aggregated across all the trees in the ensemble, resulting in a final vector $\mathbf{z}_p = [z_1, \dots, z_l]$. Each $z_i = \sum_{j=1}^t z_{ij}$ is obtained by summing the decision values for the i -th label along all the trees, as illustrated in Figure 2b. Subsequently, similarly to neural network classifiers, the softmax (2) is applied to \mathbf{z}_p , and the class corresponding to the highest probability in $\sigma(\mathbf{z}_p)$ is designated as the predicted class for the input sample \mathbf{p} . Our intuition is to use GRADIENT BOOSTING decision values as the LOGITS vector produced by neural networks.

5.2 MAXLOGIT computation

The MAXLOGIT open-set recognition module \mathcal{O} operates as a binary classifier, employing the maximum LOGIT value obtained from \mathcal{C} to distinguish between *novel* and *not novel* classes. When a sample \mathbf{p} is put into the GRADIENT BOOSTING classifier \mathcal{C} , we extract the LOGITS vector $\mathbf{z}_p = [z_1, \dots, z_l]$ and identify the maximum value $\max(\mathbf{z}_p) = \max(z_1, \dots, z_l)$. To decide whether a sample belongs to a novel class, we apply a threshold τ to $\max(\mathbf{z}_p)$ and, if $\max(\mathbf{z}_p) < \tau$, it indicates that the classifier is not confident in its classification, thereby we classify \mathbf{p} as *novel*:

$$\mathcal{O}(\mathbf{z}_p) = \begin{cases} \text{Novel} & \text{if } \max(\mathbf{z}_p) < \tau \\ \text{Not novel} & \text{otherwise.} \end{cases}$$

The value of the threshold τ is fundamental to control false alarms raised by the OSR module, *i.e.*, the amount of samples belonging to *not novel* malware families classified as instances of a *novel* family. Since the classification of a sample \mathbf{p} as belonging to a new family leads to a subsequent manual inspection by a human expert, the tuning of threshold τ is essential to avoid waste in human time resources. Threshold τ is typically tuned using an external training set $\mathcal{TT} = \{(\mathbf{p}_i, \ell_i) \mid \ell_i \in \mathcal{L}\}_{i=1, \dots, k}$, which is composed only of samples belonging to known classes. This is due to the fact that the false positive rate is solely impacted by misclassifications of known samples as novel. In particular, given a trained closed-set classifier \mathcal{C} , an external training set \mathcal{TT} and a desired false positive rate $FPR \in [0, 1]$, we set the threshold τ such that:

$$\mathcal{P}(\max(\mathbf{z}_p) < \tau \mid \mathbf{p} \in \mathcal{TT}) \leq FPR$$

where \mathcal{P} is the probability symbol. In practice, it is enough to set the threshold τ equal to the FPR -quantile of the empirical distribution of $\max(\mathbf{z}_p)$ computed from the elements of \mathcal{TT} .

We remark that our method does not add any complexity to the GRADIENT BOOSTING closed-set classifier \mathcal{C} . The open-set recognition module \mathcal{O} is seamlessly integrated into the classification workflow, preserving an inference time complexity of $O(td)$ per sample,

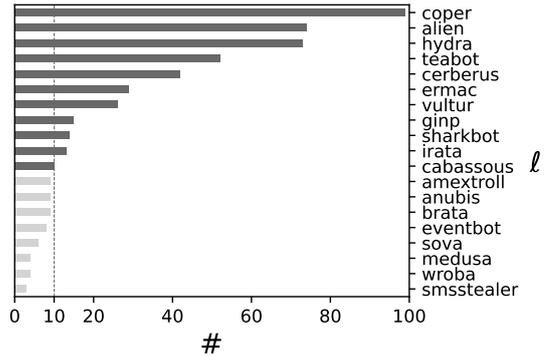


Figure 3: Imbalance in the cardinality $\#$ of each malware family ℓ in our proprietary dataset, where malware families with less than 10 samples are represented by pale-colored bars.

where t is the number of trees in the ensemble and d is the maximum tree depth.

6 Experiments

In this section, we assess the benefits of our open-set recognition solution in identifying new malware families on both a publicly available dataset and a proprietary dataset provided by our industrial partner. We first introduce the datasets in Section 6.1, and detail the experimental methodology in Section 6.2. Subsequently, we validate our approach for the open-set recognition task on the considered datasets, and discuss its performance within our industrial partner’s real-world deployment settings in Section 6.3.

6.1 Dataset

The public dataset used is Drebin [1], which consists of $n = 5560$ applications from 179 different malware families, collected in the period from August 2010 to October 2012. Each application is characterized by several features grouped into the following categories: *hardware components, required permission, app components, filtered intents, restricted API calls, used permission, suspicious API calls and network addresses*. Since this study focuses on system permissions, we considered only features in the *required permission* group explicitly referring to the Android operating system.

The proprietary dataset, kindly provided by our industrial partner, comprises records of $n = 499$ malicious applications, identified as a subset of the threat intelligence telemetry provided by an online cybersecurity software. The considered telemetry contains applications identified as potentially malicious on user devices worldwide, with a particular focus on Europe, in the period from February 2022 to January 2023. The malware labels were assigned through manual analysis of malware programs conducted by a highly specialized threat intelligence team with strong domain expertise. Figure 3 illustrates the various malware families present in the proprietary dataset and their cardinalities, highlighting the imbalanced nature of the classification problem.

Each application is described by a binary vector $\mathbf{p}_i \in \{0, 1\}^P$, where P is 154 and 1800 for the public and proprietary dataset respectively, containing the one-hot encoding of the requested Android permissions, along with the corresponding label $\ell_i \in \mathcal{L}$ denoting the malware family. The resulting datasets are in the form of $D = \{(\mathbf{p}_i, \ell_i) \mid \ell_i \in \mathcal{L}\}_{i=1, \dots, n}$.

6.2 Methodology

We evaluate the closed-set recognition performance of GRADIENT BOOSTING, on both public and proprietary datasets, through stratified 10-fold cross-validation and average results over the 10 test folds. Malware families with less than 10 samples were excluded from the 10-fold validation procedure, therefore they are not considered in closed-set recognition performance assessment. We grouped these samples in a dummy class labeled as *others*, which we then subsequently employed as the *novel* class for evaluating the open-set recognition performance of our solution. This results in an open-set recognition problem with 54 and 11 known malware families for Drebin and proprietary dataset respectively, while the *others* class includes 125 and 8 families respectively. We perform an additional experiment on the public dataset, where we grouped all the malware families other than the top 10 most populous ones into the *others* class, and we refer to this configuration as Drebin₁₀. This setup allows us to investigate a different scenario, where the closed-set classifier \mathcal{C} has to deal with a low number of populous families, while the open-set recognition module \mathcal{O} has to identify a larger, more heterogeneous set of samples as *novel*.

We perform an additional experiment following a leave-one-class-out approach, by training our model on all the populous classes of Drebin₁₀ except a single malware class, which we consider *novel* at test time. By doing so, we evaluate the effectiveness of our model in recognizing each malware class as novel when trained on the other classes. We do not consider leave- k -class-out procedures with $k > 1$, as increasing k reduces the number of known classes, thereby simplifying both the closed-set and open-set recognition tasks. This is further supported by the comparison of results between Drebin and Drebin₁₀, which differ significantly in the number of known classes. We employ stratified 10-fold cross-validation within each leave-one-class-out iteration, and average the results over the 10 test folds. To further assess the capability of our solution in identifying each individual malware class as *novel*, we plot the ROC curves for each novel class detection problem, treating the novel malware class as the positive class and merging all the classes used for training into the negative class.

Unfortunately, we have no external training set to estimate the threshold τ , nor can we use a portion of our datasets \mathcal{D} solely for this purpose due to their limited size. Therefore, we set a desired false positive rate $FPR = 0.005$ and tuned the threshold τ on *training* data, aware that by doing so we are underestimating the real false positive rate at test time. A very low FPR is mandatory in our settings, as classifying a sample p as belonging to a new family triggers manual inspection, which incurs a significant human resource cost.

6.3 Results and discussion

We first assess the effectiveness of the tree-based GRADIENT BOOSTING classifier \mathcal{C} for closed-set recognition on the 10-fold experiment on both public and proprietary datasets. A good performance in closed-set recognition is closely tied to the reliability of the subsequent open-set recognition procedure [24]. Subsequently, we assess the effectiveness of our open-set recognition classifier \mathcal{K} by comparing it against OSNN. In the latter experiment, we further investigate the relation between misclassified samples in the closed-set recognition task and the false positive rate in the open-set recognition task. Ultimately, we discuss the performance of our solution within our industrial partner’s anti-fraud business environment since its deployment on their engine.

Table 1: Micro-average (accuracy) and macro-average recall for closed-set \mathcal{C} and open-set \mathcal{K} classifiers on both public and proprietary datasets. The results are shown for the two tested settings (a) and (b).

(a) Less populous malware families grouped into a dummy class *others* and considered as the *novel* class.

	\mathcal{C}		\mathcal{K}	
	Micro	Macro	Micro	Macro
Drebin ₁₀	0.931	0.949	0.711	0.874
Drebin	0.823	0.856	0.772	0.839
Proprietary	0.863	0.869	0.842	0.844

(b) Each populous malware family, in turn, designated as the *novel* class in a leave-one-class-out fashion.

	\mathcal{C}		\mathcal{K}	
	Micro	Macro	Micro	Macro
Drebin ₁₀	0.935	0.953	0.858	0.874
Drebin	0.825	0.850	0.810	0.835
Proprietary	0.871	0.881	0.858	0.860

6.3.1 Malware classification

Table 1 summarizes the aggregated classification performance of the closed-set classifier \mathcal{C} . We observe similar performance levels on the public and proprietary dataset in both scenarios (a) and (b), with slightly better results on the proprietary dataset, and this could be attributed to the smaller number of families to classify (11 compared to 54). This is confirmed by the scenario where only the top 10 classes of the public dataset are retained, resulting in significantly higher micro-average and macro-average recall. We observe also that dealing with a higher-dimensional space (1800 compared to 154) does not necessarily make it easier to separate different classes, as in the Drebin₁₀ scenario \mathcal{C} achieves higher performance despite dealing with a comparable number of classes with respect to the proprietary dataset (10 compared to 11). The slightly lower micro-average recall, compared to the macro-average, is due to some misclassifications within the most populous classes, as the micro-average accounts for different class sizes. Overall, GRADIENT BOOSTING classifier proves effective in classifying malware families based on high-dimensional binary permission vectors.

6.3.2 Malware family discovery

Table 1 also reports the performance of the open-set classifier \mathcal{K} , which is, as expected, consistently lower than the closed-set classifier \mathcal{C} due to the additional challenge of recognizing novel malware families and the subsequent impact of false positives.

In Figures 4a and 4d, we show the recall confusion matrices of the open-set classifier \mathcal{K} when the instances of the 10 and 11 underrepresented classes, respectively for the public and proprietary datasets, are grouped in the *others* class and considered as *novel*. The recall confusion matrices reveal some false positives, where known malware families are misclassified as *novel*, especially for the proprietary dataset, with *cerberus* and *irata* families being the most affected. This suggests that these are classes where the closed-set classifier exhibits great uncertainty, resulting in low confidence levels in classification. We also observe that, while families of the public dataset are easily separable, as shown by the almost clean outer diagonal recall matrix, this does not hold for the proprietary dataset. Specifically, we observe a tendency to misclassify instances belonging to *alien* and *cerberus*, as evidenced by the prominent errors highlighted. We explain these results as a consequence of the fact that *alien* is a more recent version of *cerberus*, hence, we expect a substantial overlap in the permissions they require to the system.

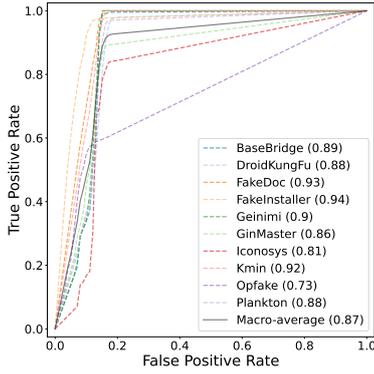
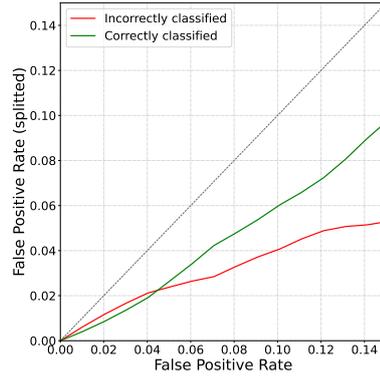
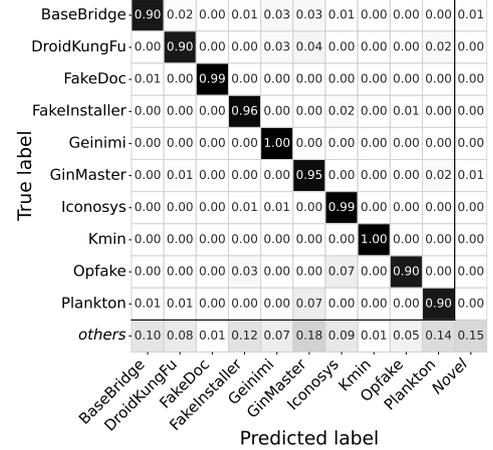


Figure 5: Novelty detection ROC curves for OSNN on the public dataset, with each class treated as *novel* in the leave-one-class-out process.



(a)



(b)

Figure 6: (a) Different contributions made by samples correctly (green curve) and incorrectly (red curve) classified by \mathcal{C} in composing the total false positive rate of the open-set recognition module \mathcal{O} on the public dataset. (b) Treating the misclassified instances as *novel* results in better real performance for the open-set classifier \mathcal{K} .

6.3.3 False alarms analysis

In this paragraph, we focus on examining the composition of false alarms on the public dataset Drebin₁₀. To this end, we decomposed the false positives of the open-set recognition module into two groups: those originating from instances that were correctly classified by \mathcal{C} before introducing the open-set recognition module \mathcal{O} , and those that would have been misclassified regardless of \mathcal{O} .

Figure 6a shows the false positive rate computed over these two groups and we observe that most of the false positives of \mathcal{O} , in particular at low *FPR* values, are samples that would have been misclassified by \mathcal{C} if the open-set module were not in place. This was expected, as instances misclassified by \mathcal{C} are typically associated with low confidence values. We argue that those samples are worth being checked by the threat analysts, since incorrect malware classification results in ineffective solutions against the threat. Therefore, if we do not consider misclassified samples as false positives, thus we compute the false positive rate only from samples belonging to known families that would have been correctly classified if there was no OSR module \mathcal{O} , the false positive rate drops considerably. Figure 6b displays the corresponding recall confusion matrix when these instances are treated as *novel*, where the model shows attains a micro-average recall of 0.714 alongside a macro-average recall of 0.877.

6.3.4 Real-world deployment performance

Our solution is currently used by analysts at our industrial partner as a complementary tool to enhance malware classification and discover new families. Testing our solution in their operational environment began in the latter half of 2023, and is currently in use in monitoring telemetry gathered by the threat intelligence division, for the analysis of data from worldwide sources.

The closed-set classification performance of our deployed solution resembles closely those reported in experiments on the proprietary dataset, achieving an accuracy of 83% when tested on about 300 telemetry applications. Unfortunately, the portion of the telemetry data used for testing did not allow us to obtain a final judgment on the model’s performance in detecting new families. The ability

to discover new real malware families cannot be assessed on the deployed system by means of strategies like the leave-one-class-out we adopted before, and recently there have been no cases of completely different malware families within the Android landscape. However, there have been a couple of noteworthy cases. In the first case, the OSR system reported as *novel* a new version of a known malware. In this case, the malware detected as *novel* had a few significant differences with respect to the other, but not enough to consider this to belong to a new family (contrary to *alien* and *cerberus*). In the second case, a variation of a known malware was labeled as *novel* due to different permissions requirements compared to the known one. Additionally, a manual review revealed that hundreds of known malware instances classified as *novel* actually belonged to families not represented in the training set, indicating they were correctly identified as *novel*. In fact, most of the false alarms concerned malware programs other than banking malware programs (the primary interest for our industrial partner’s system), and therefore correctly classified as *novel*. Our industrial partner prefers not to disclose further information on deployment performance for strategic reasons.

7 Conclusion and future works

In this study, we combined for the first time a tree-based GRADIENT BOOSTING classifier with the MAXLOGIT open-set recognition technique to tackle the problem of malware family discovery. We conducted comprehensive experiments on both a public and a proprietary dataset to validate the suitability of our approach and discussed its deployment performance in a real-world environment. Furthermore, our analysis on false alarms and the impact of misclassified instances emphasized the practical value of an open-set recognition approach in malware classification.

We envisage several potential future directions. This includes advanced feature engineering techniques to enhance the performance of open-set recognition further. Another interesting direction involves developing a dynamic thresholding system that adjusts the *FPR* threshold τ adaptively, taking into account contextual information such as the current threat landscape to achieve the desired sensibility of the OSR module.

References

- [1] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *Network and Distributed System Security (NDSS)*, volume 14, pages 8024–8035. Internet Society, 2014.
- [2] A. Bendale and T. E. Boult. Towards open set deep networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572. Institute of Electrical and Electronics Engineers (IEEE), 2016.
- [3] P. Bodesheim, A. Freytag, E. Rodner, M. Kemmler, and J. Denzler. Kernel null space methods for novelty detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3374–3381. Institute of Electrical and Electronics Engineers (IEEE), 2013.
- [4] P. Bodesheim, A. Freytag, E. Rodner, and J. Denzler. Local novelty detection in multi-class recognition problems. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 813–820. Institute of Electrical and Electronics Engineers (IEEE), 2015.
- [5] Q. Da, Y. Yu, and Z.-H. Zhou. Learning with augmented class by exploiting unlabeled data. In *Conference on Artificial Intelligence*, volume 28. Association for the Advancement of Artificial Intelligence (AAAI), 2014.
- [6] P. Faruki, V. Laxmi, A. Bharmal, M. S. Gaur, and V. Ganmoor. Androsimilar: Robust signature for detecting variants of android malware. *Journal of Information Security and Applications (JISA)*, 22:66–80, 2015.
- [7] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi. Generative openmax for multi-class open set classification. In *British Machine Vision Conference (BMVC)*. British Machine Vision Association (BMVA), 2017.
- [8] J. Guo, S. Guo, S. Ma, Y. Sun, and Y. Xu. Conservative novelty synthesizing network for malware recognition in an open-set scenario. *Transactions on Neural Networks and Learning Systems (TNNLS)*, 34(2): 662–676, 2023.
- [9] J. Guo, H. Wang, Y. Xu, W. Xu, Y. Zhan, Y. Sun, and S. Guo. Multimodal dual-embedding networks for malware open-set recognition. *Transactions on Neural Networks and Learning Systems (TNNLS)*, 36(3):4545–4559, 2025.
- [10] M. Hassen and P. K. Chan. Scalable function call graph-based malware classification. In *Conference on Data and Application Security and Privacy (CODASPY)*, pages 239–248. Association for Computing Machinery (ACM), 2017.
- [11] M. Hassen and P. K. Chan. Learning a neural-network-based representation for open set recognition. In *International Conference on Data Mining (ICDM)*, pages 154–162. Society for Industrial and Applied Mathematics (SIAM), 2020.
- [12] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer Nature, 2009.
- [13] N. Herron, W. B. Glisson, J. T. McDonald, and R. K. Benton. Machine learning-based android malware detection using manifest permissions. In *Hawaii International Conference on System Sciences (HICSS)*, pages 6976–6985. Institute of Electrical and Electronics Engineers (IEEE), 2021.
- [14] J. Jia and P. K. Chan. Representation learning with function call graph transformations for malware open set recognition. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. Institute of Electrical and Electronics Engineers (IEEE), 2022.
- [15] K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations (ICLR)*, 2018. URL <https://openreview.net/forum?id=ryiAv2xAZ>.
- [16] T. Lu and J. Wang. Domr: Toward deep open-world malware recognition. *Transactions on Information Forensics and Security (TIFS)*, 19: 1455–1468, 2024.
- [17] P. R. Mendes Júnior, R. M. De Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. De Almeida, O. A. Penatti, R. d. S. Torres, and A. Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, 2017.
- [18] M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, and R. Maskeliunas. Android malware detection: A survey. In *International Conference on Applied Informatics (ICAI)*, pages 255–266. Springer Nature, 2018.
- [19] P. Rovelli and Ý. Vigfússon. Pmds: permission-based malware detection system. In *International Conference on Information Systems Security and Privacy (ICISSP)*, pages 338–357. Springer Nature, 2014.
- [20] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(7):1757–1772, 2012.
- [21] Statista. Number of available applications in the google play store from december 2009 to december 2023, 2024. URL <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>. [Online; accessed 30-April-2025].
- [22] Statista. Global market share held by mobile operating systems from 2009 to 2023, by quarter, 2025. URL <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/>. [Online; accessed 30-April-2025].
- [23] T. N. Turnip, A. Situmorang, A. Lumbantobing, J. Marpaung, and S. I. Situmeang. Android malware classification based on permission categories using extreme gradient boosting. In *International Conference on Sustainable Information Engineering and Technology (SIET)*, pages 190–194. Association for Computing Machinery (ACM), 2020.
- [24] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman. Open-set recognition: A good closed-set classifier is all you need? In *International Conference on Learning Representations (ICLR)*, 2022. URL <https://openreview.net/forum?id=5hLP5JY9S2d>.
- [25] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets. In *Network and Distributed System Security (NDSS)*, volume 25, pages 50–52. Internet Society, 2012.