Connection Science

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ccos20

An extrinsic random-based ensemble approach for android malware detection

Nektaria Potha , V. Kouliaridis & G. Kambourakis

To cite this article: Nektaria Potha , V. Kouliaridis & G. Kambourakis (2020): An extrinsic random-based ensemble approach for android malware detection, Connection Science, DOI: 10.1080/09540091.2020.1853056

To link to this article: https://doi.org/10.1080/09540091.2020.1853056

CONNECTION SCIENCE
https://doi.org/10.1080/09540091.2020.1853056

An extrinsic random-based ensemble approach for android malware detection

Nektaria Potha[a], V. Kouliaridis [a] and G. Kambourakis [b]

[a]Department of Information & Communication Systems Engineering, University of Aegean, Karlovasi Samos, Greece; [b]European Commission, Joint Research Centre (JRC), Ispra, Italy

ABSTRACT

Malware detection is a fundamental task and associated with significant applications in humanities, cybersecurity, and social media analytics. In some of the relevant studies, there is substantial evidence that heterogeneous ensembles can provide very reliable solutions, better than any individual verification model. However, so far, there is no systematic study of examining the application of ensemble methods in this task. This paper introduces a sophisticated Extrinsic Random-based Ensemble (ERBE) method where in a predetermined set of repetitions, a subset of external instances (either malware or benign) as well as classification features are randomly selected, and an aggregation function is adopted to combine the output of all base models for each test case separately. By utilising static analysis only, we demonstrate that the proposed method is capable of taking advantage of the availability of multiple external instances of different size and genre. The experimental results in AndroZoo benchmark corpora verify the suitability of a random-based heterogeneous ensemble for this task and exhibit the effectiveness of our method, in some cases improving the hitherto best reported results by more than 5%.

ARTICLE HISTORY
Received 20 September 2020
Accepted 15 November 2020

KEYWORDS
Mobile malware; malware analysis; machine learning; ensemble learning; android security

1. Introduction

According to recent reports (Statcounter, n.d.; Statista, n.d.), the Android operating system exhibits a market share of more than 75%, making it the most widespread mobile platform worldwide (*Smartphone market share*, n.d.). As a result, countless malicious applications (apps) are deployed every year to exploit the popularity of this platform (*Mobile threat report 2020*, n.d.; *Sophos 2020 threat report*, n.d.). A great mass of mobile malware detection systems and methodologies lean towards static anomaly-based techniques, which employ machine learning (ML) to identify malicious apps (Damopoulos et al., 2014; Kouliaridis et al., 2020; La Polla et al., 2013; Narudin et al., 2016; Yan & Yan, 2018). Specifically, anomaly-based techniques comprise a training and a detection phase. The detection phase aims to unveil anomalies, i.e. deviations from the metrics obtained during the training phase. Concerning performance, static analysis requires less resources, and therefore is faster than

CONTACT G. Kambourakis georgios.kampourakis@ec.europa.eu, gkamb@aegean.gr European Commission, Joint Research Centre (JRC), 21027 Ispra, Italy

dynamic analysis. Furthermore, static analysis does not require a mobile device or a Virtual Machine (VM) to run the app, thus it is straightforward to implement.

State-of-the-art works either compare multiple classification models to determine the best performer, or use ensemble methods, which combine multiple classifiers to obtain better predictive performance. The work at hand goes one step further by proposing a dynamic ensemble selection method that concentrates on the most effective models for each malware detection case separately. That is, our method is able to take advantage of the case when multiple external malware instances are available and notably enhances state-of-the-art performance. In particular, we present an extensive study of one dynamic method based on the best performing model and meticulously study its properties and performance. This method enriches the information that is kept in each iteration when building the random subspace ensemble. Based on an extensive experimental study using benchmark datasets that cover several malware genres, and degrees of difficulty, we show that the proposed method is more effective in most of the cases. Furthermore, we demonstrate the effect of a random subspace of features in the performance of the proposed models.

A typical malware detection problem includes a set of malware cases (or instances), all derived by the same dataset or a mixed set collected by various corpora to build the positive class. A set of benign instances are sampled to construct the negative class. A malware detection method should be able to decide whether or not the instance under examination is a malware case or benign. Apart from a binary (yes/no) answer, malware detection methods usually produce a score in [0,1] that can be viewed as a confidence estimation. Essentially, malware detection problem can be defined as a one-class classification task since only labelled samples from the positive class are available. However, there are extrinsic approaches adopted that attempt to transform it to a binary classification task by sampling the negative class, i.e. all benign instances selected by other sources. In most cases, the negative class can be huge and heterogeneous since it compromises all other possible benign apps. It is important to mention that the performance of such methods heavily depends on the quality and properties of the collected external benign instances.

In this study, we handle the malware detection problem from another point of view. In particular, we build predefined positive and negative classes composed by a set of malware and goodware apps, respectively. Moreover, we treat each test instance (either malware or benign) separately by building a random subspace ensemble where in a fix number of iterations randomly choosing a subset of features and a subset of external instances (either of the positive or negative class). In each repetition, the output of three classification binary algorithms is aggregated and a negative or positive answer is provided. Experimental results on the AndroZoo benchmark dataset (Allix et al., 2016) examining different sizes and genre of external instances demonstrate the effectiveness of the proposed extrinsic approach especially in challenging cases where the set of available external instances is of limited size and in cross-genre conditions with respect to the test set.

The main contributions of this study are:

- We adopt predefined categories of external Android malware and benign instances and propose a more sophisticated extrinsic ensemble approach, which provides a positive or negative answer by averaging the output of the base models for each test instance separately. It is demonstrated that ensemble models can further improve the performance of each individual base classifier.

- We examine the effect of external instances when an ensemble malware detection method is provided combining different sizes and types of external instances. It is demonstrated that ensembles based on a larger and possibly homogeneous size of external instances are exceptionally effective alternative to ensembles included smaller sizes and feasibly more heterogeneous external instances.
- We investigate the effect of using either the entire feature set or a random subspace of features of instances in each iteration and it is demonstrated that the latter assists an extrinsic malware detection ensemble to further augment its effectiveness.
- We report experimental results on contemporary benchmark datasets and directly compare them against state-of-the-art methods under the same settings. The performance of the method presented in this study is quite competitive to the best results reported so far for these datasets, demonstrating that an extrinsic ensemble method is much more reliable and effective for the malware detection task.

The rest of this article is organised as follows. In the next section, a brief review of relevant malware detection studies is presented. Section 3 describes the examined malware detection method, while Section 4 details on the dataset employed and the experimental setup. Section 5 focuses on the performed experiments, while Section 6 discusses the main conclusions drawn from this study and elaborates on possible future work directions.

2. Related work

As of today, the topic of mobile app classification via the use of ML has received significant attention in the Android security literature (Geneiatakis et al., 2018; Kouliaridis et al., 2020; La Polla et al., 2013; Odusami et al., 2018; Papamartzivanos et al., 2014; Souri & Hosseini, 2018; Yan & Yan, 2018). This section offers a chronologically arranged review of the most notable and recent works on this topic. Specifically, as given in Table 1, we concentrate on contributions published over the last six years, that is, from 2014 to 2020. We only consider highly relevant works to ours, namely those which propose or employ some type of ensemble learning.

Yerima et al. (2015) contributed an approach which uses ensemble learning for Android malware detection. According to the authors, their method combines advantages from static analysis with ensemble learning to improve detection accuracy. During the evaluation phase, the authors collected apps from McAfee's internal repository (*Mcafee*, n.d.) and their results showed that the proposed method is capable of achieving 97.3 –99% detection accuracy with low false positive rates.

Coronado-De-Alba et al. (2016) presented an approach which analyses data obtained through static analysis. According to the authors their results provided explicit evidence for classification improvement. Even more, a comparative analysis of various ensembles were presented to find the best combination of classifiers based on the evaluation of their classification results.

Idrees et al. (2017) presented *PIndroid*, a framework which uses permissions and intents in conjunction with ensemble learning to identify Android malware. The authors evaluated their approach by applying it to 1745 real world apps from a number of datasets, including

Table 1. Outline of the related work.

Work	ML methodology	Dataset
Yerima et al. (2015)	Ensemble learning	McAfee's internal repository
Coronado-De-Alba et al. (2016)	Ensemble learning	Drebin
Idrees et al. (2017)	Ensemble learning	Contagio, Genome, theZoo, MalShare, VirusShare
Milosevic et al. (2017)	Ensemble learning	M0Droid
Chakraborty et al. (2020)	Ensemble clustering and classification	Drebin
Kouliaridis et al. (2020)	Ensemble learning	Drebin, VirusShare, AndroZoo

(*Android malware genome project*, n.d.; *Contagio mobile*, n.d.; *Malshare project*, n.d.; *thezoo aka malware db*, n.d.; *Virus share*, n.d.). Their results showed an accuracy score of 99.8%.

Milosevic et al. (2017) presented two ML-aided approaches for static analysis of Android apps. The first one is based on permissions and the other on source code analysis based on a "bag-of-words" representation model. The authors evaluated both these approaches using base classification models, as well as ensemble learning along with various combinations of the selected base models. Their results showed an *F*-score of 95.1% and *F*-measure of 89% for the source code-based and permission-based classification models, respectively.

Chakraborty et al. (2020) presented Ensemble Clustering and Classification (EC2), an algorithm for identifying Android malware families. Furthermore, the authors offered a performance comparison of several classification and clustering algorithms on the Drebin dataset (Arp et al., 2014) and used the output of both supervised classifiers and unsupervised clustering to design EC2. Their experimental results on both the Drebin and other more recent malware datasets showed that EC2 is able to accurately detect malware families, outperforming several comparative baselines. According to the authors, EC2 presents an early warning system for new malware families, as well as a predictor of known families to which a malware sample belongs.

Kouliaridis et al. (2020) introduced *Androtomist*, a novel tool capable of utilising both static and dynamic analysis on Android apps. The authors concentrated on the results of static analysis when combined with dynamic instrumentation. Moreover, they proposed an ensemble approach by averaging the output of several base models for each malware instance separately. Finally, the authors evaluated their work against three well-known datasets and their results designated that *Androtomist* is superior to previous state-of-the-art mobile malware detection solutions.

3. Methodology

To handle challenging malware detection cases, we propose essentially a random subspace ensemble taking into consideration a set of multiple malware and benign test instances ($Test_{instances}$). We coin this method *Extrinsic Random-based Ensemble (ERBE)*, and describe it in Algorithm 1. ERBE examines each test instance ($Test_{instance}$) separately. That is, within an iterative process, a subset of classification features stemming from static analysis, say, permissions, intents, etc., along with a subset of available malware and goodware samples (they are called as external instances, $Externals_{repetition}$) are randomly selected in each *repetition*. Note that exactly the same number of $Externals_{repetition}$ both from malware or

benign ($External_{malware}$ and $External_{benign}$) samples are considered per repetition to serve as a positive and negative class, respectively. The *classificationScore* of a $Test_{instance}$ with both the selected $Externals_{repetition}$ (both malware and benign) samples is calculated in terms of three classification algorithms. That is, in each *repetition*, three scores ($Score_{instance}$) of a test sample are recorded. Then, a malicious detection *MalwareDetectionScore* is calculated based on an aggregation function that combines all scores corresponding to the test sample for a number of iterations. Note that the set of classifiers applied to estimate malware detection scores per test instance and the aggregate function that combines the scores of all three classifiers for a number of repetitions can be selected among several alternatives to optimise performance in a set of preliminary experiments taken place.

As shown in Algorithm 1, the proposed method has two important parameters, $Externals_{repetition}$, and the *rate*. The former determines the size of the set of the selected external instances of both two categories. Always, an equal number of positive and negative instances are selected either from $External_{malware}$ or $External_{benign}$. The latter parameter affects the number of selected features considered for all instances (either positive or negative) examined. If it is set equal to 1, the entire set of features is utilised to represent each instance vector per iteration in order to provide the final answer (the final *MalwareDetectionScore*). On the other hand, if it is set equal to 0.5 then exactly a half amount of the initial set of features is randomly selected and used to represent a malware instance examined. In more detail, when there is exactly a fixed *rate* equal to 0.5, then a random 50% of the initial feature set is considered within the set of all investigated instances. As concerns the number of base learners (i) applied to estimate the $Score_{instance}(i)$ per iteration can also be used as a significant parameter of ERBE method. With reference to Algorithm 1, the time complexity of ERBE is in the order of $\mathcal{O}(n^2)$. Note also that the proposed approach is a stochastic algorithm since it makes some random choices of features as well as both positive and negative instances for each tested sample.

Data: $Test_{instances}$, $External_{malware}$, $External_{benign}$
Parameters: *repetitions*, $|Externals_{repetition}|$, *rate*
Result: *MalwareDetectionScore*
for *each* $Test_{instance} \in Test_{instances}$ **do**
  Set $Score(Test_{instance}) = 0$;
  repeat*repetitions* times
    Select $Externals_{repetition} \subset External_{malware}$ randomly;
    Select $Externals_{repetition} \subset External_{benign}$ randomly;
    Select *rate* % of features randomly;
    $Score_{instance}(i)$ = ClassificationLearner ($Test_{instance}$, $Externals_{repetition}$, learner(i));
    ClassificationScore = aggregate ($Score_{instance}(:)$);
  end;
  $Score(Test_{instance})$ = *Score* ($Test_{instance}$) + *ClassificationScore* / *repetitions*;
end
MalwareDetectionScore = aggregate (*Score*(:));

Algorithm 1: The proposed *Extrinsic Random-based Ensemble method.*

4. Experimental study

4.1. Description of data

In this paper, we considered a benchmark corpora, namely AndroZoo (Allix et al., 2016) built in the framework of malware detection task. This is a widely used and continuously spreading real-world collection of Android apps selected from assorted sources, including the official Google Play app market (*playstore*, n.d.). Particularly, the collection of AndroZoo apps we used in the context of this work is dated from 2017 to 2020 and enclosed 1K malware apps, each of which has been cross-examined by a large number of antivirus products. It is important to note that AndroZoo is a challenging corpora since it includes new and more sophisticated malware samples in comparison to other datasets, including VirusShare and Drebin. We also chose a set of 1K benign apps from Google Play.

External cases: As already pointed out in Section 3, given that the proposed method follows the extrinsic paradigm, it needs a set of external both malware and benign instances per each examined test instance. In this way, we follow the practice of constructing two categories to collect such a set of instances. In particular, we use a set of 800 malware cases contained in AndroZoo dataset to compose the positive category. This set of instances is randomly selected from the initial set of 1K samples in our partial AndroZoo dataset. Following the same strategy, the negative category is constructed by a randomly selected subset of 800 benign apps from the initial 1K benign cases. It should be noted that the external instances either $External_{malware}$ or $External_{benign}$ are unique and not duplicated in test set, so that they do not affect the performance scores of base models.

Feature Selection: Static analysis was performed on all the apps mentioned in Section 4.1 using the open-source tool Androtomist (Kouliaridis et al., 2020). Specifically, each app was decompiled to get the *Manifest.xml* file and log permissions and intents to create a feature vector. Each vector is a binary representation of each distinct feature. For example, think of two apps, app1 and app2. The first uses permissions p1, p2, and intent i1, while the latter uses permissions p1, p3, and intents i2, i3. This leads to the 6-dimensional feature vector (p1, p2, p3, i1, i2, i3), and thus the feature vectors for these two apps will be (1, 1, 0, 1, 0, 0) and (1, 0, 1, 0, 1, 1), respectively. Typically, the analysis of a real-world app yields a far more lengthy vector. Precisely, the analysis of the largest set of malware and benign apps used in our experiments, that is, 1 K malware instances along with all of the 1 K benign apps collected from Google Play, produced 1002-dimensional feature vectors.

4.2. Experimental setup

As already pointed out, the full dataset used comprises a collection of 1K malware apps randomly selected by AndroZoo corpus. Moreover, 1 K benign apps were downloaded by the Google Play to comprise the negative category. The test set is constructed by randomly selecting 200 malware and 200 benign individual apps by this initial set of positive and negative instances, respectively. As already mentioned, the remaining apps per category, i.e. 800 malware and 800 benign apps, compose an initial pool of $External_{malware}$ and $External_{benign}$ samples, respectively. This way, external instances (either malware or benign) are unique and distinct with respect to the specific instances included within the test set.

As per Algorithm 1, the set of external samples ($Externals_{repetition}$) is constructed by randomly selecting a fixed number of k samples per iteration from this initial pool

of $External_{malware}$ and $External_{benign}$ instances, respectively. As can be observed from Algorithm 1, the ERBE method has several parameters that need to be tuned for a particular dataset. In this paper using AndroZoo benchmark data, all these parameters were tuned based on the training set.

Specifically, to simplify and make this process more efficient, we attempt to reduce independent parameters by setting fixed parameter values. In particular, we focus on fine-tuning parameter $k = |Externals_{repetition}|$ by selecting $k \in 50, 100,\ldots, 300$ external samples per repetition optimising the performance in the training set. Then, we set $repetitions = 5$ based on some preliminary tests. A slight, though not consistent, difference with respect to the performance of $repetitions = 15$ and $repetitions = 5$ was noticed. As concerns the features selected, in this paper, we explore two options. The first one is to use exclusively the entire set of features by setting $rate = a = 1$. This indicates that the entire vector of each instance examined is considered per iteration. The second option is to fix $rate$ of $a = 0.5$ indicating that a percent of 50% of the initial set of features are selected in each repetition. Again, doing some preliminary testing, considering $a = 0.5$ and $a = 0.75$, a significant improvement in the effectiveness of the ERBE method in comparison to the case where $a = 1$ is observed.

Moreover, three well-known and widely used supervised ML algorithms were applied, namely Logistic Regression (LR), Multiple Layer Perception (MLP), and Stochastic Gradient Descent (SGD). In a more detailed description, the entire set of the classification algorithms employed falls under eager learning. In this category, supervised learning algorithms attempt to build a general model of the malicious instances, based on the training set. Obviously, the performance of such classifiers strongly depends on the size, quality and representative of the training data. For each classifier applied, the default values of the parameter settings are used. The idea behind this is that ERBE is based on the performance of the (selected) base models on the (selected) training instances. Thus, if the base models are tuned based on the same dataset, their output on that specific dataset could be biased. Since default models are less biased in the training dataset, each ERBE model is more reliably estimated. The general model of each eager classifier is built following the 10-fold cross-validation technique, where the original dataset is randomly partitioned into 10 equal sized sub-datasets. A single sub-dataset is retained for the testing, while the remaining 9 are used for training. This process is repeated 10 times, and each time using a different sub-dataset for testing. The results are then averaged to produce a single estimation.

Our main evaluation measure is the Area Under the Receiver Operating Characteristic (ROC) curve (AUC) that quantifies the effectiveness of an examined approach for all possible malware detection score thresholds (Fawcett, 2006). Moreover, this evaluation measure does not depend on the distribution of positive/negative instances. This is extracted by examining the ranking of malware detection scores (rather than their exact values) produced when a method is applied to a dataset. However, when one has to decide about a specific malware case, the malware detection score has to be transformed to a binary answer: either a positive (malware class) or a negative (benign class) one. To this direction, a threshold can be applied to the malware detection score – actually, the calculation of AUC is based on all possible thresholds. To set this threshold, information about the distribution of positive and negative malware detection instances is paramount. In our dataset, positive and negative instances are equally distributed. To transform malware detection scores to binary answers, we follow exactly the same evaluation procedure to achieve compatibility

of comparison with previously reported results. In a more detailed description, the set of the extracted scores based on the test instances are normalised in the interval of [0,1] per classification model per examined method. To this direction, the estimation of the threshold is set equal to 0.5. Then, all malware detection scores of the test dataset that are lower/higher than this threshold are transformed to negative/positive answers. This is in accordance to the setup of previously reported results.

Finally, we select the aggregation function used in ERBE method among average, minimum, and maximum that optimises performance in the training set. Most of the times, average is selected. Since the proposed ERBE method makes stochastic choices in each repetition, each experiment is repeated five times and we report average performance. Indicatively, using a system equipped with an Intel i5 CPU and 16 GB RAM, the classification process of the test set takes about 57 min, for $k = 200$ and $a = 0.5$. This time is increased to 117 min when taking into account the full feature set.

5. Results

The following classification performance metrics are used to achieve compatibility of comparison with stare-of-the-art methods, where TP, TN, FP, and FN represent correspondingly True Positives, True Negatives, False Positives, and False Negatives.

- Accuracy $(CA) : \frac{TP+TN}{TP+TN+FP+FN}$. The number of correctly classified patterns over the total number of patterns in the sample.
- Precision $(P) : \frac{TP}{TP+FP}$. The ratio of TP values over the sum of TP and FP.
- Recall $(R) : \frac{TP}{TP+FN}$. The ratio of TP over the sum of TP and FN.
- Area Under Curve (AUC): The higher positive-over-negative value ranking capability of a classifier.
- $F1 : 2 * \frac{P*R}{P+R}$.

To demonstrate the usefulness of legacy base models in ERBE approach, Figure 1 depicts the performance of $ERBE_{LR}$, $ERBE_{MLP}$ and $ERBE_{SGD}$ on the Androzoo corpus for varying sizes of external instances (k) when $a = 0.5$ (left) and $a = 1$ (right) used for each examined instance per iteration.

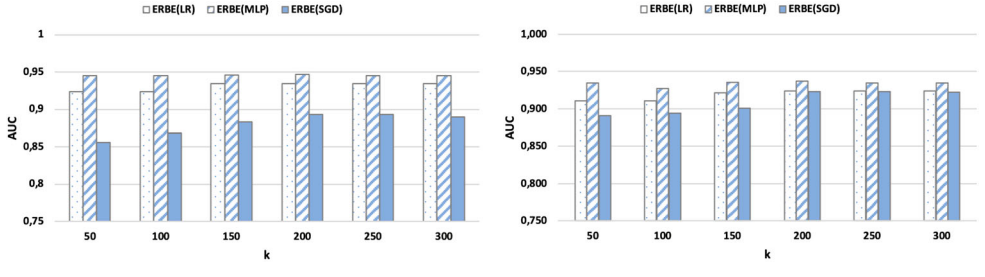

Figure 1. The performance (AUC) of the examined base models, using either $a = 0.5$ (left) or $a = 1$ (right) on AndroZoo dataset.

Table 2. Scores of all evaluation measures examined in ERBE malware detection method and the best performing base models with $a = 0.5$ and $a = 1$ for $k = 200$ on AndroZoo dataset. The best AUC score is shown in boldface.

	AUC	Accuracy	Precision	Recall	F1
ERBE$_{MLP}$, $a = 1$	0.936	0.917	0.903	0.930	0.916
ERBE$_{MLP}$	0.947	0.923	0.886	0.937	0.911
ERBE$_{a=1}$	0.978	0.950	0.940	0.959	0.949
ERBE	**0.994**	0.983	0.970	0.976	0.973

As can be clearly seen, $ERBE_{MLP}$ model is the best option in all cases of the examined size values of the external instances on Androzoo corpora. Not only in case where a percent of 50% on the initial feature set is randomly selected, but also when the entire set of features is handled, it is more effective providing a more stable performance than the two others. Moreover, the effectiveness of $ERBE_{LR}$ model is stable and seems to be competitive enough acquiring remarkable performance when $k \geq 200$ especially in both cases. On the other hand, the effectiveness of SGD classifier is negatively affected when $a = 0.5$. Then, $ERBE_{SGD}$ model seems not to be a very stable option since its performance vary for different values of k with a large margin. It seems to get improved a lot when k increases. In particular, it achieves its best results for $k = 200$, while in the case of shorter k values its performance seems to be especially poor. All these indicate that $ERBE_{SGD}$ models are in need of a more fixed and accurate set of features to be valid. Moreover, these results demonstrate that both $ERBE_{LR}$ and $ERBE_{MLP}$ are better and reliable models for multiple values of k while $ERBE_{MLP}$ is always superior.

Table 2 reports the evaluation results of ERBE method and the performance of the best base model on multiple evaluation measures over the AndroZoo dataset. Moreover, the version of ERBE when $a = 1$ and the corresponding best performing base models, $ERBE_{MLP}$ for $k = 200$, are also reported. As can be seen, ERBE is the most effective one in all cases improving always the reported results of the best base models for the specific dataset. This verifies that ensembles of classifiers based on multiple, possibly heterogeneous models, can further improve the performance of individual malware detection base models (Kamimura & Takeuchi, 2020; Liu et al., 2020; Sreeja, 2019). From the obtained results, it is also clear that the performance of ERBE is higher when a random subspace of the initial feature set is used (in case of $a = 0.5$) in comparison to the version where the entire set of features (case of ERBE$_{a=1}$) is considered in all the examined cases. It clearly seems that ERBE method is positively affected by a random rate selection of features on examined samples per iteration. In other words, when the number of features decreases and the size of vectors in examined instances is reduced then the performance of ERBE is increased. This indicates that ERBE approach is much more reliable and effective in difficult malware cases where there are irregular and incidental subsets of features which belong to different domains in each iteration.

5.1. Contribution of a random subspace set of features

Next, we examined the contribution of the factor $|a|$ related to the number of features considered in the proposed ensemble malware detection method ERBE. To isolate the contribution of this factor, beyond of the value of $a = 0.5$, we also used the value of the initial

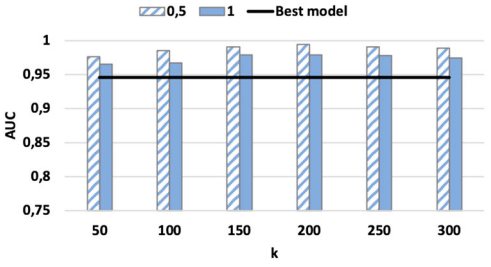

Figure 2. The performance of AUC of the proposed ERBE and ERBE$_{a=1}$. The best performing base model, $ERBE_{MLP}$ is also shown.

set of features, $a = 1$. In this way, ERBE was applied with and without considering the entire feature set for varying values of parameter k. Note that each version of the examined method was performed to the AndroZoo dataset and the aggregation function was fixed to average for this experiment.

Figure 2 shows the corresponding performance of ERBE method for comparative purposes. Apparently, the contribution of random selection of features is significant and assists the present ensemble method to enhance its effectiveness for all k values. The performance of ERBE without random collected features, $ERBE_{a=1}$ is also competitive enough, especially when the k value is increased ($k > 150$). In addition, ERBE is clearly better than the best base model, $ERBE_{MLP}$ for the whole range of the examined k values. It is noticeable that the contribution of random feature selection is stronger not only for low but also for large values of k (i.e. $k > 150$), while the version of ERBE without random feature selection (when $a = 1$) is weaker enough for relatively low values of k (up to 150). This means that k should be set to a relatively large value to reinforce the effectiveness of $ERBE_{a=1}$. In other words, this strongly suggests that when there are challenging conditions, the information of the entire feature set is less crucial for ERBE method. On the other hand, ERBE is proved to be particularly enhanced in the case where sporadic and inconstant features are selected on examined instances in each iteration.

Additionally, we examined the statistical significance of pairwise differences of both the tested versions of ERBE and base malware detection models, respectively. Table 3 demonstrates the improvement in performance (difference of AUC scores) of both the examined ERBE versions as well as the base classification models with $a = 0.5$ and $a = 1$ on AndroZoo dataset, respectively. The statistical significance of these differences is estimated using an approximate randomisation test (Noreen, 1989). The null hypothesis is that there is no difference between the two cases, and we reject this hypothesis when $p < .05$.

As can be seen, in general the models extracted from the random selected feature set are more effective and clearly better options. In particular, ERBE models based on $a = 0.5$ are more improved and gain more than the corresponding ones belonging in case of $a = 1$ for all k values. As concerns the individual base models, the results of $ERBE_{LR}$ and $ERBE_{MLP}$ are improved the most, while notable exceptions are $ERBE_{SGD}$ base models, where the random selection of features per iteration is not significantly better than the case of the entire feature set (case of $a = 1$). This does not seem to correlate with the relative increase in the number of external malware instances used per repetition.

Table 3. Improvement in performance (difference in AUC) between ensemble methods as well as base models using $a = 0.5$ and $a = 1$ on AndroZoo dataset. Statistically significant cases are shown in boldface.

k	ERBE_{LR}	ERBE_{MLP}	ERBE_{SGD}	ERBE
50	**0.024**	**0.020**	**−0.035**	0.011
100	**0.024**	**0.027**	**−0.026**	**0.019**
150	0.013	**0.021**	**−0.018**	**0.012**
200	0.010	0.011	**−0.039**	**0.016**
250	**0.023**	**0.014**	**−0.037**	**0.012**
300	**0.023**	0.011	**−0.032**	**0.015**

5.2. Comparison with the state-of-the-art

In all our experiments, the performance on the evaluation data set is measured by various evaluation measures. In this way, our reported results can directly be compared with the ones of other published methods followed the static analysis that use exactly the same evaluation measures in the framework of malware detection task. The following state-of-the-art methods, ranked in chronological order, are used to estimate the competitiveness of the proposed method:

- Yerima et al. (2015): This is an ensemble malware detection method focuses on the extraction of critical Android and Java API calls from the source code, as well as the app permissions extracted from the manifest file. In all experiments, McAfee's internal dataset is considered.
- Coronado-De-Alba et al. (2016): This method introduces a meta-ensemble algorithm. The authors employed static analysis on a dataset of 1531 malware apps collected from the Drebin dataset and 765 benign apps, to obtain permissions and intents.
- Milosevic et al. (2017): This method concentrates on the extraction of non-trivial and beneficial malicious patterns examining the usefulness of source code as well as the permissions set of features when combined with either classification or common used clustering techniques, respectively. In all experiments, the M0Droid corpus (Damshenas et al., 2015) is considered.
- Idrees et al. (2017): This is also a malware detection method based on ensemble learning to boost the effectiveness of base models followed a static app analysis. This method considers a mix set of features including app's permissions and intents derived from Contagio dump, MalGenome, theZoo, Malshare, and Virushare datasets.
- Kouliaridis et al. (2020): This is a simple heterogeneous ensemble malware detection method. A meta-model is constructed by averaging the output of several base models based on either static or hybrid analysis. The feature set comprises multiple categories such as permissions, intents and API calls. The performance of this method is evaluated on several datasets, namely Drebin, VirusShare and AndroZoo. For this work, we consider the results stemming from static analysis.

Table 4 demonstrates the effectiveness of the state-of-the-art methods per dataset on both AUC and Accuracy measures of evaluation. Note that the published results for a couple of the above methods are only provided on either AUC or Accuracy Performance measures.

Table 4. Comparison of state-of-the-art methods with the proposed ERBE malware detection method of this study (an em dash means "not applicable").

			AUC per dataset				Accuracy per dataset			
Detection method	Year	Feature set	AZ	VS	Dr	Other	AZ	VS	Dr	Other
Yerima et al.	2015	permissions, API calls	—	—	—	0.993	—	—	—	0.975
Coronado-De-Alba et al.	2016	permissions, intents	—	—	—	—	—	—	0.975	—
Milosevic et al.	2017	permissions, source code	—	—	—	—	—	—	—	0.956
Idrees et al.	2017	permissions, intents	—	—	—	0.998	—	—	—	0.998
Kouliaridis et al.	2020	permissions, intents, API calls	0.936	0.991	0.998	—	0.909	0.971	0.982	—
ERBE	2020	permissions, intents	0.994	0.993	0.997	—	0.983	0.987	0.991	—

Moreover, all these methods but the study of Coronado-De-Alba et al., are not tested on benchmark datasets, namely AndroZoo (AZ), VirusShare (VS), Drebin and (Dr), but only refer to a mixture of datasets. To the best of our knowledge, these mixed corpora are not accessible since they have generated by individual research groups and they are not publicly available. As can be seen, ERBE is particularly effective and outperforms the vast majority of baseline methods and simultaneously the study of Yerima et al. (2015), which is also based on ensemble learning. In addition, ERBE seems to be highly competitive with the approach of Idrees et al. (2017), examined on a similar feature set. However, the improved results of the proposed ERBE method in the challenging AndroZoo corpus indicate that the examined method is not easily confused in demanding malware conditions and the extracted extrinsic ensemble models can capture useful malware information.

Moreover, the improvement in performance of ERBE model with respect to that of the simple ensemble baseline as demonstrated by Kouliaridis et al. (2020) is higher than 5% on the AndroZoo dataset. All in all, given that both ERBE and simple meta-model are exclusively applied on the demanding AndroZoo dataset, it can be concluded that an extrinsic ensemble provides an effective approach in malware detection when it is fine-tuned and appropriately combined with suitable base models. With respect to the results on VirusShare and Drebin corpora, it is clear that ERBE models are very effective, competitive and even outperforming other ensemble methods. Taking into consideration the corresponding results of the methods of Coronado-DeAlba et al., and Kouliaridis et al., we notably reinforce the aforementioned outcome as our proposed extrinsic malware detection models are also very effective and clearly better options in the VirusShare and Drebin corpus. With respect to the feature types, it is clear that permissions and intents are the best option and they provide a more stable performance on ERBE in all datasets examined.

In addition, the proposed ERBE malware detection model is also very effective and clearly better option in comparison to (Milosevic et al., 2017). The improvement in performance is higher than 2%. With respect to the categories of features applied, the method of (Milosevic et al., 2017), does not seem to be positively influenced when a feature set of both app permissions and source code is available. Taking also into consideration the corresponding results of both (Milosevic et al., 2017) and (Idrees et al., 2017) methods, we notably reinforce

the aforementioned outcome as a feature set of both app's permission and intents is better able to handle challenging malware conditions.

5.3. Genre of external cases

So far, in all the experiments the set of external instances required by the examined method stem from the AndrooZoo dataset. This means that the genre of these instances most probably match a lot to the one of the test instances in question. Taking into account that the genre of all instances is the same, the performance of the proposed method can take a significant advantage and can be probably considerably improved.

This section uses another priory released corpus that will allow us to examine this effect. As concerns the set of external malware instances required by ERBE, we explored two alternatives. First, we followed the approach used in the previous experiments selecting external instances from the rather outdated Drebin corpus. We call this alternative as ERBE_{Derbin}. Second, we used the VirusShare corpus to collect the set of external instances. VirusShare comprises newer and more challenging apps dated from 2014 to 2017. This second alternative is defined as $\text{ERBE}_{VirusShare}$. For each of the above corpora examined, 200 malware apps are randomly selected per iteration by a large pool of 1 K malware samples. The benign apps included in the set of external instances are similar to those applied in ERBE and were collected from Google Play (*playstore*, n.d.). In each repetition, a set of 200 goodware apps are also randomly selected to use as external instances. Again, both the malware and benign part of the external instances was balanced with 200 positive and 200 negative cases. The parameters of the methods presented in this study were estimated based on the training part of the corpus as described in Section 4.2.

In addition, similar to the previous experiments, we examined a case of extracting the external malware instances by randomly selecting a similar number of apps from all three datasets. That way, a set of 900 mix external instances was obtained by extracting malware cases including all corpora examined (AndroZoo, Drebin, and VirusShare). In other words, in the case of external malware instances, the enriched collection comprises a mix of genres. This is called as ERBE_{mixed} alternative. Again, in each repetition, 200 malware apps are randomly collected by a mix pool of malware external instances. We call each one of the above alternatives as "genre-agnostic" because of different genre of the set of external malware instances.

Table 5 shows the results on various evaluation measures of the examined ERBE malware detection method on different genres of external malware instances selected in each iteration. From the obtained results, the best results so far are obtained by the presented ERBE model. As expected, the set of AndroZoo external malware instances assists the proposed ensemble method to achieve higher scores in comparison to cases where genre-agnostic external instances are used. This sounds reasonable since AndroZoo is a demanding corpus including new and challenging apps. In this way, it is demonstrated that the meta-learner needs as accurate base models as possible and learning models exclusively on AndroZoo samples are more likely to be more accurate than models with learning on completely different genre of external malware samples.

This is verified when genre-agnostic external malware instances are concerned since then this difference is more evident. Comparing the performance of ERBE_{Derbin} and $\text{ERBE}_{VirusShare}$ models, we see that the contribution of the latter is stronger, while the version

Table 5. Scores of all evaluation measures examined for ERBE method when $k = 200$ based on different genre of external instances. Best AUC score is shown in boldface.

	AUC	Accuracy	Precision	Recall	F1
ERBE$_{AndroZoo}$	**0.994**	0.973	0.970	0.976	0.973
ERBE$_{Drebin}$	0.971	0.938	0.936	0.940	0.938
ERBE$_{VirusShare}$	0.976	0.945	0.941	0.950	0.945
ERBE$_{mixed}$	0.987	0.941	0.925	0.960	0.942

of the ERBE$_{Derbin}$ is the most weak. This indicates that when there are cross-genre malware conditions, the information of the outdated external malware instances of Derbin corpus is less crucial on ensemble learning and inadequate to handle test instances of AndroZoo dataset. It is also remarkable that information in ensemble learning models belonging to a mix of genres can be useful to define malware cases. The ERBE$_{mixed}$ model is better than all other variations (both ERBE$_{Derbin}$ and ERBE$_{VirusShare}$ variations) in all the examined cases.

6. Discussion

In this paper, we present an extrinsic malware detection method based on ensemble learning. By utilising a set of three well-known as well as widely used base verifiers, we attempt to take advantage of their correlations by building a more sophisticated Extrinsic Random-based Ensemble (ERBE) based on a random subspace of external instances and features for each test instance separately. The experimental results based on AndroZoo benchmark dataset demonstrate that ERBE's performance is better than any single base model and is highly competitive when compared with state-of-the-art methods. The contribution of the random subspace of features, used in ERBE, is a crucial factor to improve performance. This enables ERBE to take advantage of all the examined sizes of external instances. The extrinsic ensemble approach outperforms a set of strong baselines tested on either the benchmark AndroZoo corpus or mixed datasets. The performance of ERBE is more than 5% better than an ensemble learning baseline implemented on the challenging AndroZoo dataset too. In comparison to the best baseline (note that this method is tested on a mixed dataset), ERBE is competitive enough in terms of accuracy measure.

All extrinsic methods strongly depend on the appropriate selection of external instances. In this paper, we used a set of malware instances randomly selected by AndroZoo corpus ensuring that there are similarities with the test instances under examination. Certainly, this procedure can be improved by taking into account the genre of instances. To this direction, we examined the effectiveness of ERBE method following a couple of options. In the first one, the external malware instances were exclusively collected by different malware corpora in comparison to test set. In the second, a mixed set of external malware instances derived from multiple malware corpora was considered. As it is demonstrated, ideally, the external malware instances should be sampled from the same source as the one from which the test instances are drawn. For instance, if the instance under examination belongs to AndroZoo dataset, then there is strong indication that the external instances should also be part from that dataset to ensure similarity in genre, format and edition of instances. This

can be explained since the meta-model needs as accurate and reliable base models as possible. However, information about the source of test malware instances may not be available in the most of the real-world cases.

The current work uses three classifiers as malware base models. An interesting future work direction could focus on a richer set of classification models, comprising eager and lazy algorithms, that can be adapted to each malware case separately. This heterogeneous ensemble approach relies on base models with default parameter settings. This could be used to further enrich the pool of our base verifiers considering several versions of the same approach with different fixed and tuned parameter settings. Another future work direction could concentrate on combining multiple malware detection methods based on hybrid and static analysis in a more complex approach. Lastly but not least, although in this work ERBE has been evaluated using Android malware datasets, it is evident that it can be easily applied for malware detection on any platform.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

V. Kouliaridis http://orcid.org/0000-0002-4233-5998
G. Kambourakis http://orcid.org/0000-0001-6348-5031

References

Allix, K., Bissyande, T., Klein, J., & Traon, Y (2016, May 14–22). *AndroZoo: Collecting millions of android apps for the research community*. In *Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016* (pp. 468–471), Austin, TX, United States. ACM. https://doi.org/10.1145/2901739.2903508

Android malware genome project (n.d.). Retrieved November 8, 2020, from http://www.malgenomeproject.org/

Arp, D., Spreitzenbarth, M., Gascon, H., & Rieck, K. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

Chakraborty, T., Pierazzi, F., & Subrahmanian, V. S. (2020). EC2: Ensemble clustering and classification for predicting android malware families. *IEEE Transactions on Dependable and Secure Computing*, *17*(2), 262–277. https://doi.org/10.1109/TDSC.8858

Contagio mobile (n.d.). Retrieved November 8, 2020, from http://contagiominidump.blogspot.com/.

Coronado-De-Alba, L. D., Rodríguez-Mota, A., & Escamilla-Ambrosio, P. J. (2016, November 15–17). *Feature selection and ensemble of classifiers for Android malware detection*. 8th IEEE Latin-American Conference on Communications, LATINCOM 2016, Medellin, Colombia (pp. 1–6). IEEE. https://doi.org/10.1109/LATINCOM.2016.7811605

Damopoulos, D., Kambourakis, G., Gritzalis, S., & Park, S. O. (2014). Exposing mobile malware from the inside (or what is your mobile app really doing?). *Peer-to-Peer Networking and Applications*, *7*(4), 687–697. https://doi.org/10.1007/s12083-012-0179-x

Damshenas, M., Dehghantanha, A., K. K. Choo, & Mahmud, R. (2015). M0Droid: An android behavioral-based malware detection model. *Journal of Information Privacy and Security*, *11*(3), 141–157. https://doi.org/10.1080/15536548.2015.1073510

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Geneiatakis, D., Baldini, G., Fovino, I., & Vakalis, I. (2018, February 26–27). Towards a mobile malware detection framework with the support of machine learning. In Erol Gelenbe, Paolo Campegiani, Tadeusz Czach'orski, Sokratis K. Katsikas, Ioannis Komnios, Luigi Romano and Dimitrios Tzovaras. *Security in Computer and Information Sciences - First International ISCIS Security Workshop 2018, Euro-CYBERSEC 2018, Revised Selected Papers, Communications in Computer and Information Science* London, UK (Vol. 821, pp. 119–129). Springer. https://doi.org/10.1007/978-3-319-95189-8_11

Idrees, F., Rajarajan, M., Conti, M., Chen, T. M., & Rahulamathavan, Y. (2017). PIndroid: A novel android malware detection system using ensemble learning methods. *Computers & Security*, *68*, 36–46. https://doi.org/10.1016/j.cose.2017.03.011

Kamimura, R., & Takeuchi, H. (2020). Improving collective interpretation by extended potentiality assimilation for multi-layered neural networks. *Connection Science*, *32*(2), 174–203. https://doi.org/10.1080/09540091.2019.1674245

Kouliaridis, V., Barmpatsalou, K., Kambourakis, G., & Chen, S. (2020). A survey on mobile malware detection techniques. *IEICE Transactions on Information and Systems*, *E103.D*(2), 204–211. https://doi.org/10.1587/transinf.2019INI0003

Kouliaridis, V., Kambourakis, G., Geneiatakis, D., & Potha, N. (2020). Two anatomists are better than one-dual-level android malware detection. *Symmetry*, *12*(7), 1128. https://doi.org/10.3390/sym12071128

La Polla, M., Martinelli, F., & Sgandurra, D. (2013). A survey on security for mobile devices. *IEEE Communications Surveys & Tutorials*, *15*(1), 446–471. https://doi.org/10.1109/SURV.2012.013012.00028

Liu, W., Hu, E. W.., Su, B., & Wang, J. (2020). Using machine learning techniques for DSP software performance prediction at source code level. *Connection Science*, 1–16. https://doi.org/10.1080/09540091.2020.1762542

Malshare project (n.d.). Retrieved November 8, 2020, from https://malshare.com/about.php

Mcafee (n.d.). Retrieved November 8, 2020, from https://www.mcafee.com/en-us/index.html

Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). Machine learning aided android malware classification. *Computers & Electrical Engineering*, *61*, 266–274. https://doi.org/10.1016/j.compeleceng.2017.02.013

Mobile threat report 2020 (n.d.). *Mobile threat report 2020*. (n.d.). Retrieved November 8, 2020, from https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf

Narudin, F., Feizollah, A., Anuar, N., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, *20*(1), 343–357. https://doi.org/10.1007/s00500-014-1511-6

Noreen, E. (1989). *Computer-intensive methods for testing hypotheses: an introduction*. Wiley.

Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R., & Maskeliunas, R. (2018, November 1–3). Android malware detection: A survey. In Hector Florez, Cesar Diaz, & Jaime Chavarriaga (Eds.), *Applied Informatics - First International Conference, ICAI 2018, Bogot'a, Colombia, 2018, Proceedings, Communications in Computer and Information Science* (Vol. 942, pp. 255–266). Springer. https://doi.org/10.1007/978-3-030-01535-0_19

Papamartzivanos, D., Damopoulos, D., & Kambourakis, G. (2014). A cloud-based architecture to crowdsource mobile app privacy leaks. In *Proceedings of the 18th panhellenic conference on informatics* (pp. 1–6). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2645791.2645799

playstore (n.d.). Retrieved November 8, 2020, from https://play.google.com/store

Smartphone market share (n.d.). Retrieved November 8, 2020, from https://www.idc.com/promo/smartphone-market-share/os

Sophos 2020 threat report (n.d.). Retrieved November 8, 2020, from https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophoslabs-uncut-2020-threat-report.pdf

Souri, A., & Hosseini, R. (2018). A state-of-the-art survey of malware detection approaches using data mining technique. *Human-Centric Computing and Information Sciences*, *8*, 3. https://doi.org/10.1186/s13673-018-0125-x

Sreeja, N. K. (2019). A weighted pattern matching approach for classification of imbalanced data with a fireworks-based algorithm for feature selection. *Connection Science*, *31*(2), 143–168. https://doi.org/10.1080/09540091.2018.1512558

Statcounter (n.d.). *Mobile operating system market share worldwide*. Retrieved November 8, 2020, from https://gs.statcounter.com/os-market-share/mobile/worldwide

Statista (n.d.). *Mobile operating systems' market share worldwide from January 2012 to December 2019*. Retrieved November 8, 2020, from https://www.statista.com/statistics/272698/

thezoo aka malware db (n.d.). Retrieved November 8, 2020, from https://thezoo.morirt.com/

Virus share (n.d.). Retrieved November 8, 2020, from https://virusshare.com

Yan, P., & Yan, Z. (2018). A survey on dynamic mobile malware detection. *Software Quality Journal*, *26*(3), 891–919. https://doi.org/10.1007/s11219-017-9368-4

Yerima, S. Y., Sezer, S., & Muttik, I. (2015). High accuracy android malware detection using ensemble learning. *IET Information Security*, *9*(6), 313–320. https://doi.org/10.1049/iet-ifs.2014.0099