POSTER: Accuracy vs. Time Cost: Detecting Android Malware through Pareto Ensemble Pruning

Lingling Fan[†], Minhui Xue[†‡], Sen Chen[†], Lihua Xu[†], Haojin Zhu[♮]
[†]East China Normal University, Shanghai, China
[‡]NYU Shanghai, Shanghai, China
[♮]Shanghai Jiao Tong University, Shanghai, China

ABSTRACT

This paper proposes *Begonia*, a malware detection system through Pareto ensemble pruning. We convert the malware detection problem into the bi-objective Pareto optimization, aiming to trade off the classification accuracy and the size of classifiers as two objectives. We automatically generate several groups of base classifiers using SVM and generate solutions through bi-objective Pareto optimization. We then select the ensembles with highest accuracy of each group to form the final solutions, among which we hit the optimal solution where the combined loss function is minimal considering the trade-off between accuracy and time cost. We expect users to provide different trade-off levels to their different requirements to select the best solution. Experimental results show that *Begonia* can achieve higher accuracy with relatively lower overhead compared to the ensemble containing all the classifiers and can make a good trade-off to different requirements.

Keywords

Malware Detection; Ensemble Pruning; Pareto Bi-objective Optimization; Begonia

1. INTRODUCTION

Mobile devices have become the potential target of attackers due to the massive downloads of applications in recent years. Malicious applications that illegally obtain private information or perform harmful actions to the devices pose a severe threat in our daily life. Recent approaches tried to alleviate this problem to achieve high detection accuracy by applying machine learning. For example, DREBIN [1] extracted thousands of features for machine learning and achieved high accuracy in malware detection. Smutz *et al.* [5] applied ensemble learning to malware detection, which improved the true positive rate by detecting poor classifiers and providing a confidence in the prediction of ensemble classifiers to indicate that the classifier is not fit to provide an accuracy respond. Ensemble learning is also applied to hardware-supported malware detection [3]. However, these approaches only focus on the detection accuracy, but neglect the computational cost.

CCS'16 October 24-28, 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4139-4/16/10.
DOI: http://dx.doi.org/10.1145/2976749.2989055

In this paper, we propose a malware detection system through Pareto ensemble pruning to trade off the classification accuracy and the computational cost. Pareto ensemble pruning [4] formulates this classification problem as a bi-objective optimization problem, with accuracy and computational cost as the two objectives. We take the vote of the predictions to automatically generate pruned ensembles. Since the prediction time of the ensembles is in the unit of millisecond level, we further evaluate the performance with respect to the training time of different groups (different sizes of base learner pools) and trade off the accuracy and the time of ensemble pruning process. Furthermore, we select the ensemble of the highest accuracy in each group to form the Pareto solutions, among which users are required to provide a trade-off level (*i.e.*, weight) to obtain the best required solution where the combined loss function is minimal. Experimental results show that our detection system, termed *Begonia*, can achieve relatively higher accuracy with relatively fewer learners compared to the ensemble containing all the base classifiers.

In summary, we make the following contributions:

- We propose a customized trade-off between accuracy of malware classification and computational cost.
- We provide an automatic Pareto ensemble pruning framework for malware detection.

2. BEGONIA ARCHITECTURE

The high level execution process of our approach, as shown in Figure 1, occurs in four phases: (i) *Reverse engineering*, which prepares resource files for extracting features by decompiling the APKs; (ii) *Feature Extraction*, which extracts features from each application using both static and dynamic analysis; (iii) *Ensemble pruning*, which trains large sets of labeled Android applications to obtain several groups of base classifiers containing n classifiers ($n = 10, 20, \ldots$) and selects the base classifiers guided by bi-objective optimization, to trade off accuracy and computational cost; and (iv) *Classification*, which classifies the dataset into different categories, benign and malicious, based on the optimal pruned ensembles.

We provide an ensemble selection approach based on Pareto optimal to trade off accuracy and computational cost. Since the prediction time of the ensembles is so small compared to the pruning time, we only evaluate the performance with respect to the pruning time, and trade off the accuracy and the time of ensemble pruning process. We train different numbers of base learners for each group using SVM, and select the ensemble with highest accuracy via Pareto ensemble pruning. Obviously, the more base learners it trains, the more time it will take to prune the ensemble with highest accuracy for each group. The selected ensembles are then provided to the decision makers to select an optimal solution considering

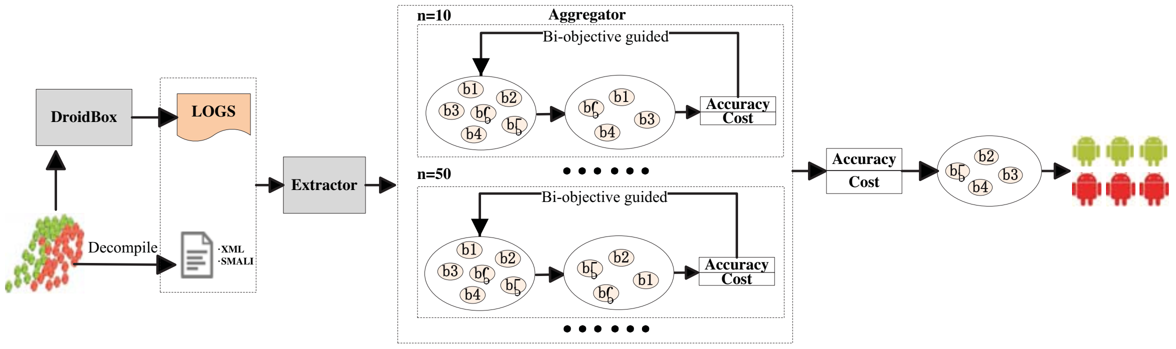

Figure 1: Overview of *Begonia*

other requirements and information. Compared to one-objective optimization that usually has only one optimal solution, Pareto optimal has multiple optimal solutions, and all the solutions have redeeming features. They cannot be simply excluded. Pareto optimal is introduced to explain the dominant relation between each solution. The Pareto domination relation is then formalized as follows.

DEFINITION 1. *(Pareto Domination)*
Let $h = (h_1, h_2) : S \to \mathbb{R}^2$ denote the objective vector mapping from the solution space S to $\mathbb{R}^2$. For two solutions $u, v \in S$, u dominates v iff it meets the following conditions:
(1) $h_1(u) \le h_1(v)$ and $h_2(u) \le h_2(v)$
(2) $h_1(u) < h_1(v)$ or $h_2(u) < h_2(v)$
(For simplicity, vectors will not be denoted by boldface characters in this paper.)

A solution is Pareto optimal if there does not exist a solution that can be better without sacrificing some of the other objective values. More specifically, a solution u is Pareto optimal if there is no other solution in S that dominates u. The solutions are provided to the decision makers to select an optimal solution considering other requirements and information.

Ensemble pruning, also known as ensemble selection, selects a subset of classifiers from base classifiers set and classifies a new dataset by taking the vote of their predictions. It tries to achieve the goal that the accuracy of the combining prediction results improves and in parallel the computational overhead reduces compared to the ensemble containing all the classifiers. Consider the number of the base learners in $B_m = \{b_1, \ldots, b_m\}$ is m, and let T_t denote a pruned classifier set with the selected vector $t \in \{0,1\}^m$, where $t_i = 1$ indicates the base learner b_i is selected for the ith component. The optimal pruned ensemble $T_{opt.sel}$ can be formulated as follows:

$$T_{opt.sel} = \underset{t \in \{0,1\}^m}{\arg\min}\ E(T_t) + w \cdot |T_t|,$$

where $E(T_t)$ is the validation error rate of T_t. Since the measurement of generalization classification performance is hard to declare, we use the validation error on the validation date set instead. Given a validation dataset with k instances, for validation instance i, $T_t(x_i)$ is the prediction value of T_t, and y_i is the actual value. $E(T_t)$ is calculated as

$$E(T_t) = \frac{1}{k}\sum_{i=1}^{k} \chi(T_t(x_i) \neq y_i),$$

where $\chi(\cdot)$ is the indicator function, which equals 1 if the expression holds; otherwise, it equals 0, $|T_t| = \sum_{i=1}^{m} t_i$ is the size of the selected learners, $w \in [0, +\infty]$ is the trade-off level, and $E(T_t) + w \cdot |T_t|$ is the combined loss function aiming to obtain the solutions that minimize the combined loss to achieve good performance.

Consider there are several trade-off levels to be solved, the optimal pruned ensemble $T^{(i)}_{opt.sel}$ can be defined based on different trade-off levels w_i, for all i:

$$T^{(i)}_{opt.sel} = \underset{t \in \{0,1\}^m}{\arg\min}\ E(T_t) + w_i \cdot |T_t|.$$

ALGORITHM 1: Customized Ensemble Pruning

```
Input: The training set $N_1$, the validation set $N_2$, and Trade-off levels
    $W = \{w_1, w_2, \ldots, w_l\}$.
Output: Pruned ensembles $\mathcal{T} = \left\{T^{(1)}_{opt.sel}, T^{(2)}_{opt.sel}, \ldots, T^{(l)}_{opt.sel}\right\}$.
1: Let $f(T_t) = (E(T_t), |T_t|)$ denote the bi-objective function
2: $B_{10} = g(N_1), B_{20} = g(N_1), \ldots, B_m = g(N_1)$
3: $\mathcal{I} = \{\text{bi-objective-solver}(f(T_t))\}$
4: $P = \{p_1, p_2, \ldots, p_q\}$
   $P$: the ensemble with the highest accuracy of each group
5: for $j = 1$ to $l$ do
6:     $T^{(j)}_{opt.sel} = \arg\min_{T_t \in P}\ E(T_t) + w_j \cdot |T_t|$
7: end for
8: return $\mathcal{T} = \left\{T^{(1)}_{opt.sel}, T^{(2)}_{opt.sel}, \ldots, T^{(l)}_{opt.sel}\right\}$.
```

As shown in Algorithm 1, the ensemble pruning process takes as input the training dataset, the validation set, and outputs the pruned ensembles that minimize the combined loss with w_i. $g(N_1)$ denotes that the training set for each base learner is randomly chosen from N_1 in an out-and-in manner. The base classifiers of different groups are generated on different training sets using SVM. We implement the PEP algorithm [4] to solve the bi-objective optimization problem, and return a Pareto optimal ensemble set $\mathcal{I}$ for each group. The idea behind PEP solver is that it randomly selects an ensemble from the base classifier pool, denoted as $t \in \{0,1\}^m$, stores it in P, and flips each bit with probability $\frac{1}{m}$ and generates t'. The goal is to evaluate if there exist solutions in P that dominate t'. If it is true, it continues this process without augmenting P; otherwise it excludes the ones that are dominated by t' and adds t' into P. From the second selection on, it selects an ensemble each time from P, flips, and generates another ensemble. This process is iterative until it reaches the upper-bound of iteration times. Finally, P contains the Pareto solutions. We then pick out the ensembles with the highest accuracy of each group $P = \{p_1, p_2, \ldots, p_5\}$ (see line 4 in Algorithm 1) to determine the final ensembles with different trade-off levels.

3. EMPIRICAL EVALUATION

The goal of our experiments is to examine the relation between accuracy and time cost of real-time analysis.

3.1 Dataset and Setup

The 4,000 benign samples are downloaded from Google Play Store, and the 4,000 malicious samples are from [2]. Following the methodology of [2], we select 155 features in total to perform a binary classification. Four types of features are shown below:

- *Permission*. Android required permission for each app can be extracted from the *AndroidManifest* file. It is often used as a metric to detect malware. We finally select 59 out of the original 120 permissions.
- *Sensitive API Calls*. API calls are extracted from the *smali* files that are generated by decompiling the APKs. We finally select 90 out of the 240 extracted sensitive API calls.
- *Sequence*. Sequence is extracted from *smali* files by recording the number of sensitive API calls requested by the malicious apps and the benign ones, respectively. Three quantitive metrics are applied to extract features, which are "Subtraction-Differential" metric, "Logarithm-Differential" metric, and "Subtraction-Logarithm" metric [2].
- *Dynamic Behavior*. Dynamic behavior observes the malicious activities triggered by each application through analyzing the log files of *DroidBox* [2].

We automatically generate the base classifiers on each training set using SVM, whereby we prune the base classifiers and calculate the error rate on the validation set. The size of the base classifier pool is n ($n = 10, 20, \ldots, 50$ of each group in our experiment). Iteration times is set to be $\lceil n^2 \log n \rceil$ when dealing with the bi-objective solver [4].

3.2 Accuracy vs. Time Cost

Table 1 shows that ensemble pruning takes more time to obtain an optimal ensemble when pursuing higher accuracy. To trade off the two objectives, provided by a trade-off level w, we select the final ensemble that minimizes the combined loss (*i.e.*, $E(T_t) + w \cdot |T_t|$). For example, as shown in Figure 2, given different trade-off levels, $w = 0.0006$ and $w = 0.00025$. The final ensembles are the p_2 and p_4 accordingly, minimizing the combined errors.

Table 1: Accuracy vs. Time Cost

# Group Size	Time (sec)	Accuracy
10	60	93.40%
20	460	94.20%
30	1,546	94.70%
40	3,654	95.00%
50	7,450	95.20%

Note that accuracy column indicates the highest accuracy of each group.

3.3 Discussion

(i) Dependency on bi-objective Pareto optimization solver. The ensembles selected by Algorithm 1 highly rely on the performance of the bi-objective solver. The computational complexity of the solver used is expected to $\mathcal{O}\left(k^2 \log k\right)$, indicating that the expected iterations for generating the approximating optimal Pareto set is $\mathcal{O}\left(k^2 \log k\right)$, where k is the size of the base classifier pool. Moreover, the solver [4] has been proved to be more effective than other ensemble pruning methods, thereby rendering the performance of our approach relatively reliable.

(ii) Limitations of ensemble pruning process. Since we use Bagging to obtain our base classifiers, the random out-and-in strategy of selecting the training set may cause some randomness in each training process. Different size of the training set and different iteration times of the Pareto ensemble pruning may affect the performance of the pruned ensembles. We therefore conduct our experiment n times to choose the best size and iteration times to ensure our experimental results reliable.

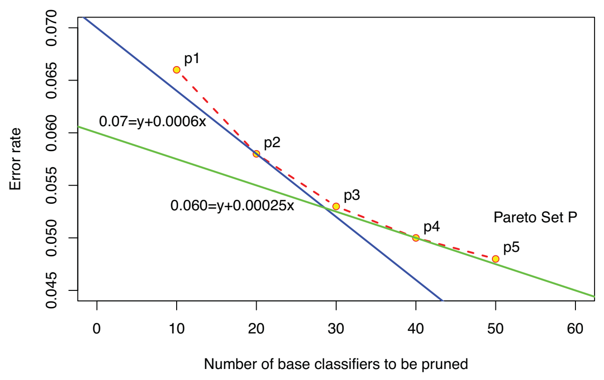

Figure 2: An example of selecting an optimal from Pareto solutions

4. CONCLUSION

In this paper, we proposed a malware detection system, termed *Begonia*, through Pareto ensemble learning to trade off classification accuracy and time cost. (We only consider the pruning time in this paper, since prediction time is so small as to be negligible.). Experimental results show that *Begonia* can trade off accuracy and time cost when given a desirable trade-off level and achieve a relatively higher accuracy with relatively lower overhead. Only time will tell whether *Begonia* can be highly effective either as a standalone system or as a complementary technique to contemporary tools to overcome the limitations of traditional anti-malware solution in detecting the zero-day and modern malware.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China, under Grant 61502170, 61272444, 61411146001, U1401253, and U1405251, in part by the Science and Technology Commission of Shanghai Municipality under Grant 13ZR1413000, and in part by Pwnzen Infotech Inc.

5. REFERENCES

[1] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*, 2014.

[2] S. Chen, M. Xue, Z. Tang, L. Xu, and H. Zhu. Stormdroid: A streaminglized machine learning-based system for detecting android malware. In *Proceedings of the 11th ACM on Asia CCS*, pages 377–388. ACM, 2016.

[3] K. N. Khasawneh, M. Ozsoy, C. Donovick, N. B. Abu-Ghazaleh, and D. V. Ponomarev. Ensemble learning for low-level hardware-supported malware detection. In *RAID*, 2015.

[4] C. Qian, Y. Yu, and Z.-H. Zhou. Pareto ensemble pruning. In *AAAI*, 2015.

[5] C. Smutz and A. Stavrou. When a tree falls: Using diversity in ensemble classifiers to identify evasion in malware detectors. In *NDSS*, 2016.