Identifying Android Malware Using Network-Based Approaches

Emily Alfs
Department of Computer Science
Kansas State University
Manhattan, KS USA
emilyalfs@ksu.edu

Doina Caragea
Department of Computer Science
Kansas State University
Manhattan, KS USA
dcaragea@ksu.edu

Dewan Chaulagain
Department of Computer Science
Bowling Green State University
Bowling Green, OH USA
dewanc@bgsu.edu

Sankardas Roy
Department of Computer Science
Bowling Green State University
Bowling Green, OH USA
sanroy@bgsu.edu

Nathan Albin
Department of Mathematics
Kansas State University
Manhattan, KS USA
albin@ksu.edu

Pietro Poggi-Corradini
Department of Mathematics
Kansas State University
Manhattan, KS USA
pietro@ksu.edu

***Abstract*—The proliferation of Android applications has resulted in many malicious apps entering the market and causing significant damage. Robust techniques that determine if an app is malicious are greatly needed. We propose the use of network-based approaches to effectively separate malicious from benign apps, based on a small labeled dataset. The apps in our dataset come from the Google Play Store and have been scanned for malicious behavior using VirusTotal to produce a ground truth dataset with labels *malicious* or *benign*. The apps in the resulting dataset have been represented in the form of binary feature vectors (where the features represent permissions, intent actions, discriminative APIs, obfuscation signatures, and native code signatures). We have used these vectors to build a weighted network that captures the "closeness" between apps. We propagate labels from the labeled apps to unlabeled apps, and evaluate the effectiveness of the approaches studied using the F1-measure. We have conducted experiments to compare three variants of the label propagation approaches on datasets that consist of increasingly larger amounts of labeled data.**

***Index Terms*—Classification, Semi-supervised learning, Android Malware,**

I. Introduction

As the number of Android applications becoming available and subsequently downloaded is rapidly growing, there has been a similar increase in the number of malicious software, referred to as malware, appearing on the market. Malware can have many different purposes and perhaps the most prolific is obtaining personal information. This can include medical and financial information as well as passwords, which can be used for malicious purposes. With the threat of such information getting into the wrong hands, it is important to find a solution to the Android malware detection problem.

There are many techniques that determine if an application is malicious, and they are ever-changing. Ideally, techniques to identify malware must be robust, as the schemes of creating malicious apps are changing as well. When working with Android applications, there are two main hurdles: the lack of labeled data and the imbalance of data (i.e., the number of malware apps is significantly smaller than the number of benign apps). To address the lack of labeled data, we explore semi-supervised learning approaches, specifically network-based transductive learning approaches. To address data imbalance, we explore a class mass normalization (CMN) approach.

Network-based transductive learning approaches make use of a small number of labeled instances and propagate the given labels to a typically much larger set of unlabeled instances. In our case, the instances are the apps. Each app is represented as a binary vector, where features represent permissions, intent actions, discriminative APIs, obfuscation signatures and native code signatures [1]. To construct the app network, we test several methods for evaluating the similarity between apps, including the Gaussian and the k-nearest-neighbors kernel ($k-$NN). We compare three network-based transductive learning variants: Label Propagation (a.k.a., hard clamping) and Label Spreading (a.k.a., soft-clamping), as implemented in Python through scikit-learn [2], and a variant proposed as part of our research. We also compare the transductive approaches with two supervised baselines, specifically k-nearest-neighbors (k-NN) and naive Bayes (NB).

This project is partially supported by the National Science Foundation under Grants No. 1717871 and No. 1515810. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASONAM '19, August 27-30, 2019, Vancouver, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6868-1/19/08
http://dx.doi.org/10.1145/3341161.3343534

Our contributions in the Android application domain are the following:

- We assess the impacts of standard similarity metrics (Gaussian kernel, and k-NN kernel) on our data.
- We compare transductive network-based approaches (hard clamping, soft clamping and a variation) to supervised learning techniques (naive Bayes and k-NN).
- We evaluate the robustness of these methods in the presence of data imbalance, which is prevalent in security.
- We analyze the impacts of CMN on transductive learning when the datasets used exhibit different levels of class imbalance (including no class imbalance).

II. Related Work

Many researchers have studied labeling techniques in many different applications including malware detection. In particular, machine learning has been used in malware detection by [1], [3] and many others. Social networks have also been explored in the context of malware detection [4], [5].

Furthermore, this is not the first time when semi-supervised learning, specifically, transductive learning, has been used for Android malware detection. For example, [6] used semi-supervised learning where each edge is the file delivery relation between various files, URLs, and IPs. Their proposed approach first used a supervised classifier for each node type, then used those the resulting probabilities as priors for the label propagation. They specifically used Bayesian Label Propagation, with the iterative function $f_i = \frac{\sum_{v_j \in N_i} f_j + \gamma \hat{y}_i}{\sum_{v_j \in N_i} f_j + \gamma}$, where γ is a shape parameter, $\hat{y}_i$ is the prior, $f_i = \frac{|N_i^+| + \gamma \hat{y}_i}{|N_i| + \gamma}$, and $|N_i|$ is the set of neighbors to point i with $|N_i^+|$ being neighbors with positive labels. Semi-supervised learning techniques were also used in [7]. Here, the authors defined their similarity metric as the strength of co-occurrences using the Jaccard similarity measure. In their research, they used the label propagation technique called hard-clamping [8], which we outline in our Methods section. In [9], the authors used an attributed function call graph. In this setup, each vertex represents a local function which is first translated into an intermediate language, where six types of attributes are extracted. Then, to determine distances between applications, they first performed a pairwise function distance computation, malware distance computation, then a pairwise malware similarity computation using the Gaussian kernel. Finally, a maximum confidence label propagation technique was used.

III. Methods

A. Hard Clamping (HC) Algorithm

The hard clamping algorithm described in this subsection was proposed in [8]. Let $(x_1, y_1)...(x_l, y_l)$ be our labeled data where the set of $Y_l = \{y_1, ...y_l\}$ are the class labels. The number of classes, c, is assumed to be known. In our specific case, the number of classes is 2, class 1 being benign and class 2 being malicious. We also have a set of unlabeled data, $(x_{l+1}, y_{l+1})...(x_{l+u}, y_{l+u})$, for which we are trying to assign labels. Let $Y_u = \{y_{l+1}, ...y_{l+u}\}$ be the set of unobserved labels and $X = \{x_1, ...x_{l+u}\}$ be the data points.

We define weight matrix W, where $w_{i,j} \in W$ is the weight between nodes i and j. This weight is given by the similarity of the feature vectors of the corresponding apps; thus, the more similar apps are, the larger the edge weight.

We define a $(l+u) \times (l+u)$ matrix T. This is a probabilistic transition matrix where $T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}}$. We can think of this as the probability to move from node j to node i. We also define matrix Y, a $(l+u) \times c$ matrix, where each row, i, corresponds to the probability distribution of node x_i.

We start by initializing Y for the first l nodes, which are labeled. The algorithm works as follows:

1) Propagate $Y \leftarrow TY$
2) Row-normalize Y
3) Clamp the labeled data. Repeat step 1 until Y converges.

To expand, in step 2), we row normalize Y, as each row represents a probabilistic matrix, thus each row must sum up to one. In step 3), by clamping the labeled data, we are resetting the values of Y for the labeled data back to the initial values, thus, ensuring that the nodes that have a ground truth associated with them do not change their labels. The final labeling for application i is given by the maximum of row Y_i. Reference [8] proves that this iterative method does converge.

B. Soft Clamping (SC) Algorithm

The soft clamping variant of the label propagation algorithm was proposed in [10]. Given a set of points $X = \{x_1, ..., x_l, x_{l+1}..., x_{l+u}\}$, where the first l points are labeled and the remaining are unlabeled, let Y be an $n \times c$ matrix, where n is the number of apps and c is the number of classes. Entry $Y_{i,1}$ is the probability of application i being benign and $Y_{i,2}$ the probability of application i being malware. We build an affinity matrix W, which in our case is the matrix, where the w_{ij} entry is the edge weight between apps i and j and we define $w_{ii} = 0$. By this definition, W is a symmetric matrix. We construct $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, where D is a diagonal matrix such that D_{ii} is the sum of the elements of the ith row of W. Entry-wise we have $S_{ij} = \frac{W_{ij}}{\sqrt{D_{ii} D_{jj}}}$. Let $F = [F_1^T, ..., F_n^T]^T$, where each F_i is an $1 \times c$ matrix that corresponds to the classification of app i. The algorithm then iterates

$$F(t+1) = \alpha S F(t) + (1-\alpha) Y \tag{1}$$

until convergence [10]. In this algorithm, $Y = F(0)$ represents the initial labels and α is a scalar. We can interpret α as the amount of information received from neighboring apps and the app's own initial labeling. To get the final label for app i, we look at row i of the final iteration, and the label is chosen to correspond to the column where the maximum occurs.

C. Proposed Variant for Label Propagation

As mentioned, the final algorithm we used is a combination of both soft and hard clamping. We build the matrix $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ as outlined in the soft clamping algorithm. We also define Λ as a diagonal matrix with the same dimensions

as S, where $\lambda_{ii} = \alpha_i$. Note that α_i corresponds to the i-th app and $0 \leq \alpha_i \leq 1$ for all i. For apps with highly accurate initial labels, such as apps that were manually verified, we use a low α value, while we use a high α value for the unlabeled apps. Specifically, for apps with highly accurate labels, we use $\alpha = 0$, while for unlabeled apps $\alpha = 0.9$. Thus, we are ensuring that hard clamping is used for the quality ground truth data, and soft clamping for the unlabeled data. Our variant is based on the iterative equation from soft clamping:

$$F(t+1) = \Lambda S F(t) + (I - \Lambda) Y \tag{2}$$

Similar to soft clamping, we proved that this iterative method will converge as well. To determine the label of app i, we look at row i of F at convergence, and the label is chosen to correspond to the column where the maximum occurs.

D. Class Mass Normalization (CMN)

For each of the methods, upon convergence, the label of each app is determined by the maximum of the two column values for that app. Alternatively, one can use the CMN technique, introduced in [8]. Intuitively, CMN takes into account the class imbalance ratio for labeled data and imposes that imbalance on the unlabeled data after propagation. Working with a domain where class imbalance is prevalent, we can apply CMN to mitigate the imbalance.

Further explanation of CMN is available in [11]. Let $y_{i,k}$ be the initial label of app i and let p_k be the prior probability of class k based on the labeled data. We define p_k as $p_k = \frac{1}{l} \sum_{i=1}^{l} y_{i,k}$. The average of estimated weights for class k on the unlabeled data is given by $m_k = \frac{1}{u} \sum_{i=l+1}^{l+u} \hat{y}_{i,k}$ where $\hat{y}_{i,k}$ is the class produced by hard clamping. Then, the class normalization occurs by taking $\arg\max_k w_k \hat{y}_{i,k}$ where $w_k = \frac{p_k}{m_k}$. Using this evaluation, we take into account the imbalance of data. We compare the label propagation methods with CMN and without CMN, for imbalanced as well as balanced data.

E. Similarity Matrix

Label propagation methods rely on an affinity matrix to define how similar applications are to one another. Results are highly dependent on the choice of the similarity metric. We tested three different similarity metrics, specifically, the k-NN and Gaussian kernels, and the similarity metric derived from the Hamming distance. We found in initial trials (results not shown) that the Gaussian kernel performed the best. Results and other resources suggested that this is due to the fact that the Gaussian kernel is much smoother than k-NN and the Hamming distance. Therefore, the Gaussian kernel was used to produce the results in this work. The Gaussian kernel for apps x and y is defined as $K(x, y) = exp(-\gamma ||x - y||^2)$, where γ is a parameter that needs to be tuned.

IV. Experimental Setup and Results

A. Experimental Data

The set of apps in this work was curated by [1] this set contains over 1.3 million apps. Associated with each app in the set is a binary feature vector with 471 entries which represent various features. The features fall into the following categories: permissions, intent actions, discriminative APIs, obfuscation signatures, and native code signatures. In our base dataset, an app is labeled malware or benign based on the number of VirusTotal [12] programs that predict the app to be malware.

B. Research Questions

Our experiments are designed to answer the following questions:

Research Question 1: How do the transductive label propagation techniques compare to each other, and also to standard supervised classifiers, specifically k-NN and naive Bayes (NB)?

Research Question 2: How do the label propagation techniques perform in the presence of class imbalance?

Research Question 3: How does the CMN technique impact results in both balanced and imbalanced data cases?

C. Experimental Setup

Each method that is being studied has respective parameters which need to be tuned. Given a dataset, we selected parameter values based on the best performance of a method on that particular dataset. To estimate the performance of the methods, we calculated the average F-measure over five random labeled/unlabeled splits. The F-measure is given by $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$.

In order to find good parameter values for each method, we performed a grid search for both the k-NN kernel and the Gaussian kernel. For the k-NN kernel, we used the following values for k: $\{3, 5, 7, 9, 11\}$. For the Gaussian kernel, we used the following values for γ, as suggested by [13]: $\{2^{-15}, 2^{-13}, ..., 2^3\}$. Results showed that any γ less than 2^{-3} performed poorly in all methods. Thus, the following results are obtained using $\gamma \geq 2^{-3}$. For soft clamping and the proposed variant, α must also be tuned. We used the following values for α: $\{0.1, 0.2, ..., 0.9\}$.

We trained the approaches studied on datasets that contain increasingly larger labeled data, as shown in Table I for balanced data in the upper portion and unbalanced in the lower. A smaller labeled dataset is a proper subset of a larger labeled dataset.

We read the table as follows: in row one, our first dataset, we started with 50 labeled malicious and 50 labeled benign apps, as well as 5,000 'unlabeled' malicious and 5,000 'unlabeled' benign apps. By 'unlabeled', we are referring to the fact that we know the ground truth for these applications but we have the program treat them as though they are unknown, allowing us to evaluate the performance of the approaches studied. We trained the transductive label propagation techniques on the graph constructed from both labeled and unlabeled apps. We evaluated the performance on the unlabeled apps. To compare with supervised classifiers, we trained the models on the labeled data and evaluated them on the unlabeled data.

TABLE I
NUMBER OF LABELED AND UNLABELED INSTANCES IN THE EXPERIMENTS WITH BALANCED (UPPER) AND UNBALANCED (LOWER) DATA

Labeled Malware	**Labeled Benign**	**Unlabeled Malware**	**Unlabeled Benign**
50	50	5000	5000
100	100	5000	5000
500	500	5000	5000
1000	1000	5000	5000
5000	5000	5000	5000
10	100	500	5000
25	250	500	5000
50	500	500	5000
250	2500	500	5000

D. Experimental Results

We outline the results from the testing of each algorithm: hard-clamping (HC), soft-clamping (SC) and the variation (VAR). We compare these against each other and also against well-known supervised algorithms, k-nearest-neighbors and naive Bayes. Table II (upper), shows the average F1-measure of the best tuned models, over five splits, for balanced datasets.

Comparing the transductive label propagation techniques, we found that hard-clamping performed the best. With the exception of the iteration with 100 labeled malware applications, the transductive label propagation techniques did perform better than the supervised counterparts. As can be seen, the results show that CMN did not offer consistent improvements for balanced data.

In the case of imbalanced data, the class ratio was 1:10, and the results of the best tuned models over five data splits are shown in Table II (lower). In comparison to the balanced case, the average F1-measure values are lower for imbalanced data. Comparing the transductive label propagation techniques, we found that the soft-clamping technique and the variant performed almost equally and better than hard-clamping. This does change, however, when we apply CMN. We see a significant improvement in performance for hard-clamping after implementing CMN, making the hard-clamping technique with CMN the best method overall. As can be seen from the table, the CMN technique did help in the case of imbalanced data. With the exception of the smallest dataset, k-NN performed better than the transductive label propagation techniques without CMN. However, as mentioned above, this was not the case after CMN was applied to the transductive label propagation techniques.

TABLE II
F1-MEASURE RESULTS FOR BALANCED (UPPER) AND UNBALANCED (LOWER) DATASETS.

Data size	NB	k-NN	HC	HC CMN	SC	SC CMN	VAR	VAR CMN
50	0.771	0.858	0.879	0.878	0.872	0.876	0.871	0.876
100	0.873	0.931	0.907	0.908	0.902	0.899	0.899	0.899
500	0.803	0.937	0.940	0.939	0.937	0.937	0.937	0.937
1000	0.811	0.941	0.945	0.945	0.946	0.945	0.945	0.945
5000	0.866	0.958	0.965	0.964	0.965	0.964	0.965	0.965
10	0.456	0.343	0.602	0.716	0.622	0.640	0.622	0.639
25	0.532	0.749	0.680	0.793	0.714	0.720	0.714	0.721
50	0.518	0.783	0.748	0.843	0.760	0.793	0.760	0.794
250	0.526	0.907	0.831	0.918	0.847	0.897	0.847	0.897

V. CONCLUSIONS AND FUTURE WORK

In this study, we compared several transductive label propagation techniques on the Android malware identification problem. We performed experiments on both balanced and imbalanced datasets. We used the Gaussian kernel to construct the app graph. In the balanced case, we found that hard-clamping performed the best out of all of the methods which were tested. As intuition suggests, CMN lead to no significant improvements in the models for balanced data. For the imbalanced data with ratio 1:10, CMN offered significant improvements. We found that hard-clamping with CMN was the best approach overall among those that we compared.

For future work, we would like to study the impacts of noise on the performance by artificially introducing noise into our data. The ground truth for Android app detection is generally noisy, thus this is a worthwhile task.

REFERENCES

[1] S. Roy, J. DeLoach, Y. Li, N. Herndon, D. Caragea, X. Ou, V. P. Ranganath, H. Li, and N. Guevara, "Experimental study with real-world data for android app security analysis using machine learning," in *Proc. of the 31st Annual Computer Security Applications Conference*, ser. ACSAC 2015. New York, NY, USA: ACM, 2015, pp. 81–90.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. DeLoach, D. Caragea, and X. Ou, "Android malware detection with weak ground truth data," in *2016 IEEE Int. Conf. on Big Data (Big Data)*, Dec 2016, pp. 3457–3464.

[4] M. Ni, T. Li, Q. Li, H. Zhang, and Y. Ye, "Findmal: A file-to-file social network based malware detection framework," *Knowledge-Based Systems*, vol. 112, pp. 142–151, 2016.

[5] L. Chen, W. Hardy, Y. Ye, and T. Li, "Analyzing file-to-file relation network in malware detection," in *Web Information Systems Engineering – WISE 2015*, J. Wang, W. Cellary, D. Wang, H. Wang, S.-C. Chen, T. Li, and Y. Zhang, Eds. Cham: Springer Int. Publishing, 2015, pp. 415–430.

[6] I. Alabdulmohsin, Y. Han, Y. Shen, and X. Zhang, "Content-agnostic malware detection in heterogeneous malicious distribution graph," in *Proc. of the 25th ACM Int. Conf. on Information and Knowledge Management*, ser. CIKM '16. New York, NY, USA: ACM, 2016, pp. 2395–2400.

[7] M. Ni, Q. Li, H. Zhang, T. Li, and J. Hou, "File relation graph based malware detection using label propagation," vol. 9419, 11 2015, pp. 164–176.

[8] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Computer Science, CMU, Tech. Rep., 2002.

[9] D. Kongand G. Yan, "Transductive malware label propagation: Find your lineage from your neighbors," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 1411–1419.

[10] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schlkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004, pp. 321–328.

[11] Y. Bengio, O. Delalleau, and N. L. Roux, "Label propagation and quadratic criterion," in *Semi-Supervised Learning*, O. Chapelle, B. Schölkopf, and A. Zien, Eds., 2006, pp. 193–216.

[12] VirusTotal, 2018. [Online]. Available: https://www.virustotal.com/

[13] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," May 2016. [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide