# Droid-Sec: Deep Learning in Android Malware Detection

Zhenlong Yuan[‖‡] Yongqiang Lu[†] Zhaoguo Wang[≀] and Yibo Xue[‡*]
[†]Baidu Inc., Beijing, China
[‖]Department of Automation, Tsinghua University, Beijing, China
[‡]Research Institute of Information Technology, Tsinghua University, Beijing, China
[*]Tsinghua National Lab for Information Science and Technology, Beijing, China
[≀]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
yuanzl11@mails.tsinghua.edu.cn, luyongqiang@baidu.com, {wangzhaoguo, yiboxue}@tsinghua.edu.cn

## ABSTRACT

As smartphones and mobile devices are rapidly becoming indispensable for many network users, mobile malware has become a serious threat in the network security and privacy. Especially on the popular Android platform, many malicious apps are hiding in a large number of normal apps, which makes the malware detection more challenging. In this paper, we propose a ML-based method that utilizes more than 200 features extracted from both static analysis and dynamic analysis of Android app for malware detection. The comparison of modeling results demonstrates that the deep learning technique is especially suitable for Android malware detection and can achieve a high level of 96% accuracy with real-world Android application sets.

## Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and protection*

## Keywords

Android malware, deep learning, detection

## 1. INTRODUCTION

A recent report from Gartner[1] shows that Android tablet sales grew 127 percent and reached the No.1 position in 2013. Application market such as Google Play Store is playing an important role in the popularity of Android devices and drive the economy of Android applications. However, the openness of the Android market also makes it a hot target for malware attacks, which is a serious threat for network users' security and privacy. Consequently, there is an urgent need to pick up the malware from normal apps.

Currently, the main defense mechanism for Android to fight against malware is a risk communication mechanism, which will warn users about the permissions an app requires before the user installs it. This approach is indeed ineffective as it presents the permissions of an app in a "stand-alone" fashion and requires too much technical knowledge for a general user to distinguish malware. Note that a normal app and a malicious app may require the same permissions and thus it is hard for users to make a right decision. In fact, more users tend to know directly whether it is a malware or not, without concerning too much on the risk assessment.

Deep learning [1], as a new area of machine learning research, has gained increasing attentions in artificial intelligence (AI) and motivated a great number of successful applications in speech and image recognition. In this paper, we first extract more than 200 features[2] from both static analysis and dynamic analysis of each Android app, and then apply the deep learning technique to classify the malware from normal apps. In addition, our comparison of modeling results demonstrates that the deep learning technique is far more suitable than some other machine learning techniques, includes Naïve Bayes, SVM, C4.5, Logistic Regression (LR) and Multi-layer Perceptron (MLP).

## 2. STATIC AND DYNAMIC ANALYSIS

In order to systematically characterize Andorid apps (i.e. normal and malicious apps), we employ both static and dynamic analysis to extract 202 features from each app. In particular, these features will fall into three types: required permission, sensitive API and dynamic behavior. Among them, required permission and sensitive API are extracted through static analysis while dynamic behavior is extracted through dynamic analysis, as shown in Table 1. Note that some other types of features can also be added in our model to make it better for characterizing Android apps [3]. Here our main goal is to demonstrate that deep learning can achieve a high level of accuracy with a set of key features.

In the case of static analysis, all we need is the .apk file of an Android app. After uncompressing the apk file with the '`7-Zip`' tool, we mainly focus on parsing the two files 'AndroidManifest.xml' and 'classes.dex' respectively. By parsing the 'AndroidManifest.xml' file with the tool '`AXMLPrinter2`' and the parser '`TinyXml`', we can know what permissions an app indeed requires, e.g. permission '`android.permission.call_phone`' stands for permitting an app to make a phone call. Actually, we totally search for 120 permissions in this part. Besides, by parsing 'classes.dex' file with the disassembler '`baksmali`', we can know what sensitive API will be called, e.g. function '`ContentResolver;->delete`' stands for a sensitive API and might be used for

[*]Corresponding author.
[1]`http://www.gartner.com/newsroom/id/2674215`

[2]More details can be found at http://www.droid-sec.com.

*SIGCOMM'14,* August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631434.

**Table 1: Features from Analysis**

| Static Analysis | | Dynamic Analysis |
|---|---|---|
| **Required Permission** | **Sensitive API** | **Dynamic Behavior** |
| ACCESS_FINE_LOCATION | IActivityManager*Stub*Proxy;->shutdown | ACTION_DEXCLASS_LOAD |
| ACCESS_COARSE_LOCATION | ActivityManager;->killBackgroundProcesses | ACTION_RECVNET |
| ACCESS_MOCK_LOCATION | ActivityManagerNative;->restartPackage | ACTION_SERVICESTART |
| . . . . . . | . . . . . . | . . . . . . |

deleting users' messages or contacts. Totally, we search for 64 sensitive APIs in this part.

In the case of dynamic analysis, we run the .apk file of an app in the 'DroidBox[3]'. DroidBox is a kind of sandbox that is developed based on TaintDroid [2]. After running an app for a fixed period of time, DroidBox can generate some information log which includes the dynamic behaviors that happened from the app, such as '`action_sendnet`' which stands for an action that sends data over the network. Totally, we search for 18 dynamic behaviors in the part.

## 3. DEEP LEARNING MODEL

Traditional machine learning models, such as SVM etc. that has less than three layers of computation units, are considered to have shallow architectures. Fortunately, deep learning with a deep architecture changed that. In reality, deep learning model actually can be trained in various ways with different approaches or algorithms. In this paper, due to the space limitation, we mainly present our framework on building model for Android malware detection as shown in Figure 1.

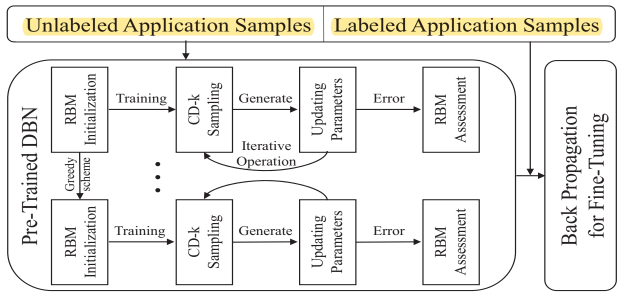

**Figure 1: Framework of Deep Learning Model**

As a semi-supervised training algorithm, deep learning consists of two phases, the 'unsupervised pre-training phase' and the 'supervised back-propagation phase'. In the pre-training phase, we adopt the deep belief network (DBN) [1] for pre-training. The DBN is hierarchically built by stacking a number of restricted Boltzmann machines (RBM) with regarding the deep neural network as a latent variable model, which is beneficial for better characterizing Android apps. In the back-propagation phase, the pre-trained neural network is to be fine-tuned with labeled value in a supervised manner. After that, the whole deep learning model is built completely. Besides, more details will be presented in Section 4.

## 4. EVALUATION

To validate our deep learning model in Android malware detection, we experiment on public application sets (mixed with malware apps and normal apps). The malware set (250 samples) is downloaded from the famous *contagio mobile*[4] and the normal app set is crawled from the top 250 apps in *Google Play Store*. Note that the number of malware and the number of normal apps are always mixed equally in the training or test sets.

**Table 2: Accuracy with Different Constructions**

| Num of Layers | Num of Neurons | Accuracy |
|---|---|---|
| 6 | [150,150,150,150,150,150] | 94.0% |
| 5 | [150,150,150,150,150] | 95.0% |
| 4 | [150,150,150,150] | 94.5% |
| 3 | [170,170,170] | 93.0% |
| **3** | [150,150,150] | 96.5% |
| 3 | [130,130,130] | 95.0% |
| 2 | [150,150] | 89.5% |

Significantly, there are two key parameters while building the deep learning model, one is the number of layers in DBN and the other is the number of neurons in each layer. Table 2 shows the accuracy changes with different model constructions. In addition, we conduct a comparison between the deep learning model and other five typical machine learning models as shown in Table 3. The five machine learning models are all optimized for the best accuracy by using the grid search technique.

**Table 3: The Comparison of Modeling Results**

| Model | Training Set | Test Set | Accuracy |
|---|---|---|---|
| SVM | 300 | 200 | 80.0% |
| C4.5 | 300 | 200 | 77.5% |
| Naïve Bayes | 300 | 200 | 79.0% |
| LR | 300 | 200 | 78.0% |
| MLP | 300 | 200 | 79.5% |
| **Deep Learning** | 300 | 200 | 96.5% |

To better mitigate mobile malware threats, we plan to automate the deep learning based tool for online Android malware detection in our future work, you can follow up our ongoing project at http://www.droid-sec.com.

## 5. ACKNOWLEDGEMENT

This work was supported by the National Key Technology R&D Program of China under Grant No.2012BAH46B04.

## 6. REFERENCES

[1] Y. Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[2] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI'10*, volume 10, pages 1–6, 2010.

[3] Y. Zhou and X. Jiang. Dissecting android malware: characterization and evolution. In *IEEE S&P'12*, pages 95–109, 2012.

[3] `https://code.google.com/p/droidbox`

[4] `http://contagiominidump.blogspot.com`