Mini Review over “DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks”

Author: Zhijie Liu, Student ID: 2022233365

1. What is the research problem, and what is the significance of the research?

In recent years, with the increasing popularity of mobile devices, more and more application developers have been developing mobile apps with comprehensive functions for users. But as apps come into people's lives, emerging mobile malware bring a lot of uncertainty to social security, such as remote control, economic and financial fraud, privacy data leakage and so on. Therefore, ensuring the security of published apps has become necessary.

Android is the most popular mobile operating system, but its open source makes it the target of 97% of malware. The ever-changing malware makes their malicious behaviors complex and diverse. Moreover, with sophisticated code obfuscation technologies, malware can be changed greatly at the code structure level but still retaining its original semantic behaviors, reducing the code malicious density. Besides, analysts in security companies are faced with a large number of malware waiting to be analyzed every day, which is a very time-consuming and labor-intensive task. Therefore, it is necessary to propose automated Android malware detection method for scanning malware.

2. What is state-of-the-art research status of the research problem?

DL-based detection has gradually become the most effective detection methods due to that there is a large amount of data available. Traditional signature-based methods are not robust and generalizable enough since their customized security mode. Yet, ML-based methods are heavily dependent on the hand-crafted engineering.

DL-based detection methods can be simply classified into four categories, 1) FCN-based, 2) CNN-based, 3) RNN-based, and 4) GNN/GCN-based. 1) FCN-based method, especially MLP-based, consists of at least three dense layers, and can learn non-linear patterns of data. Its detection speed is fast, but lacking accuracy. 2) CNN-based method utilizes CNN architecture for characterizing apps followed with a fully connected layer as a classifier, which can learn useful context information. However, the code pattern is sequential, it cannot accurately learn the long-term sequence dependencies. 3) RNN-based method utilizes RNN, GRU or LSTM architecture for remembering previous knowledge, which can capture a larger and more complex code context and semantics. Due to the above advantages, it performs achieving high accuracy and robustness, but costing a lone time. 4) GNN/GCN-based method applies to graphs of apps for characterizing apps. The graphs can be in many forms such as control flow graph, data flow graph, function call graph, and so on. Thus, this method can capture structure information and achieves high performance. But, it is also liable to structural attack.

3. Describe the methodology of the paper, and describe the advantage of the proposed method over state-of-the-art.

In order to resolve the effectiveness, efficiency and hand-crafted engineering problem. This paper proposes the DeepRefiner system based on two DL-based detection models, MLP-based detection model and LSTM-based detection model. The first detection layer is utilized to filter malicious, benign and uncertain apps, and the second detection layer is used to precisely detect the uncertain apps. The two-layer structure system ensures the detection efficiency and effectiveness evaluated on 62,915 malware and 47,525 benign apps, the largest scale dataset.

The first detection layer is a MLP-based detection model. It takes content in xml files in apps as features for filtering malware and benign apps. Apps that cannot be reliably detected are marked as uncertain and sent to the second detection layer for further scanning. More specifically, it takes label and attribute values in xml files (AndroidManifest.xml file and xml files in /res/), including permissions, components, intents, strings and so on. Then, all these string format features are ordered as a vector, where every coordinate represents a corresponding string format feature, placed with 1 (containing the feature) or 0 (not containing the feature). Thus, an app can be transformed into a feature vector as the input of MLP for both training and testing. Predicted label is based on the following formula:

$$\text{Label} = \begin{cases} \text{Uncertain} & \text{if } max(P_{\text{Malicious}}, P_{\text{Benign}}) < \text{threshold} \\ \text{Malicious} & \text{elif } (max(P_{\text{Malicious}}, P_{\text{Benign}}) \geq \text{threshold}) \\ & \text{and } (P_{\text{Malicious}} > P_{\text{Benign}}) \\ \text{Benign} & \text{else} \end{cases}$$

The second detection layer is a LSTM-based detection model for detecting the uncertain apps left from the first detection layer. It takes simplified smali code as input. The concrete steps of it include: 1) transfer and transform all code of dex files in one app into one file in smali format; 2) simplify all smali code; 3) utilize byte2vec embedding algorithm for transforming all smali code statements into vectors and get a semantic bytecode vector sequence (representation); 3) take the vector sequence as input into LSTM for app representation; and 4) attach a classifier for classification. These steps are included in both training and testing phases.

DeepRefiner's two-layer structure help effectively and efficiently detect unknown apps. These two advantages are difficult to contain in other methods. The first detection layer is able to rapidly filter malicious and benign apps with high accuracy, and leave unfiltered uncertain apps into the second detection layer. Due to LSTM-based detection model, the second detection layer can capture semantics among all code statements, thus, precisely distinguish malicious apps among uncertain apps. In fact, from the experiment result, over 70% apps can be classified at the first detection stage, which makes DeepRefiner achieve high performance in both time and accuracy. Moreover, DeepRefiner is robust in detecting obfuscated malware since the obfuscation of code is unable to change the semantic information.

4. What is the conclusion? On what way can one can possibly improve the performance of the method.

DeepRefiner is Android malware detection system combining two detection layers in complementary way to provide refined detection. The first detection layer, based on content in xml files, provides efficiently filtering effect for filter malicious and benign apps and leave uncertain apps into the second detection layer for further detection. The second detection layer, through capturing semantic information, detects uncertain apps based on their smali code and achieve high effective and robust. The whole design of DeepRefiner is based on abstraction. DeepRefiner detects apps through different abstraction levels (coarse-grained behavior vs. fine-grained behavior), which is actually also an assumption on the distribution of apps, representing simple versus complex.

According to the method of DeepRefiner, there are two ways may be able to improve its performance: 1) refine byte2vec embedding algorithm. Make it enable to learn the semantic information of opcodes and operands, which can enhance LSTM-based model learning malicious behaviors; and 2) introduce dynamic analysis to capture real malicious behaviors, since static analysis is difficult to resolve Java and Kotlin's dynamic properties.

5. What is the inspiration of the paper to your own research, like on writing, on theory development, on experimental design, or on research idea etc.?

After reading this paper, there are actually three aspects inspiring my own research.

① Paper Writing: This paper explains every step and point in detail, to make explicit and clear enough for paper readers. As I was writing my paper, I find that my description of many issues was not deep enough. For example, when I presented a specific method, I always neglected to explain the context used in it, missing justification. This caused that the paper was well-reasoned from my point of view, but not understandable from readers. Therefore, I must be sure to explain the causality of the content from the reader's point of view;

② Experimental Design: This paper's experiment design is quite adequate. After careful analysis of its experiment and those of other papers, I find that most of their experiments start from a coarse-grained perspective. May be, I can design an experiment from a fine-grained perspective; and

③ Research Idea: This paper, as well as current related papers, lack automated explaining of malicious behaviors, which is the shortage of DL-based detection method. However, it is not impossible to conquer.

For each question, no less than 100 words is preferred.

Words Count: 1320