Explainable Machine Learning Techniques for Program Modeling and Mobile Application Security

Android has evolved to be the most popular mobile operating system over the past several years, since its inception. Owing to its ease of use and openality, it attracts thousands of vendors and developers working on application (app) development. Millions of Android apps provide a variety of functionalities to users, such as instant messaging, gaming and online shopping. However, due to its prevalence, Android also becomes a prime attack target of cybercriminals. According to Symantec's 2016 Annual Threat Report, the number of Android malware has reached 13 million. A major reason for such tremendous volume is Android's app packaging model which offers painless and straight-forward opportunities for attackers to piggyback (i.e., inject) their malicious code on popular benign apps which could then be spread through either Google Play (official market) or other third-party markets. These malware may steal sensitive information, control devices remotely or encrypt on-device data for ransom thus putting the end users at high risk. Besides malware authors, some unscrupulous developers clone the code from benign apps and repackage it with advertisements and new functionalities, thus stealing revenue from the original developers. The sheer volume, growth rate and evolution of malware and clone apps highlight an imperative need for developing effective and scalable automated detection techniques.

To perform automated detection, recent approaches both from academia and industry increasingly resort to Program Analysis and Machine Learning (ML) techniques. Typically, the detection process involves extracting semantic features from suitable representations of programs (e.g., assembly code, call graphs) and identifying malice or clone code patterns using ML classification or clustering algorithms. However, most of these ML algorithms could be applied only on data represented as vectors. Hence, a pivotal factor in determining the effectiveness of these detection processes, is building suitable vector representations of

programs. We intend to address this as the primary motive of this thesis. Recognizing that higher level semantic representations of programs such as call graphs, control- and data-flow graphs mostly stay similar even when the code is considerably altered, we intend to learn their representations (i.e., graph embeddings) and use them to perform accurate and efficient malware and clone detection. We use a common term Program Representation Graphs (PRGs) to refer to any of the aforementioned graphs. Once appropriate PRG embeddings are built, as a secondary motive, we intend to address specific issues in malware and clone detection processes, separately. In the case of malware detection, three issues: (1) population drift induced by malware's evolution and (2) systematic integration of multi-modal multi-granular features from different sources and (3) precisely locating malicious code portions in PRGs (malice methods/classes) have been addressed. In the case of clone detection, scalability which remains as a crucial bottleneck has been addressed. The following are the achievements made in this thesis:

1. It is well-known that Android malware constantly evolves so as to evade detection. This causes the entire malware population to be non-stationary. Contrary to this fact, most of the prior works on ML based malware detection have assumed that the distribution of the observed malware characteristics (i.e., features) does not change over time. We address the problem of malware population drift and propose a novel online learning based framework to detect malware, named CASANDRA (Context-aware, Adaptive and Scalable ANDRoid mAlware detector). In order to perform accurate detection, a novel graph kernel named Contextual Weisfeiler-Lehman Kernel (CWLK) that facilitates capturing apps' security-sensitive behaviors along with their context information from PRGs is proposed. Besides being accurate and scalable, CASANDRA has specific advantages: (i) being adaptive to the evolution in malware features over time (ii) explaining the significant features that led to an app's classification as being malicious or benign. When evaluated with more than

87,000 apps collected in-the-wild, CASANDRA achieves 89.92% accuracy, outperforming existing techniques by more than 25% in their typical batch learning setting and more than 7% when they are continuously retained, while maintaining comparable efficiency.

2. Existing malware detection approaches have typically used classifiers with a variety of features such as API/instruction sequences observed, control-flow structures exhibited, privileges used and information sources/sinks used. These feature sets provide complementary perspectives (interchangeably referred as *views*) of apps' behaviors with inherent strengths and limitations. Meaning, some views are more amenable to detect certain attacks while they may not be suitable to characterize certain other attacks. For instance, information flow features are amenable for detecting '*privacy lea*k' attacks but incapable of revealing '*privilege escalations*'. Existing approaches (incl. CASANDRA) use either one or a selected few of the aforementioned feature sets which prevents them from detecting a substantial majority of attacks. To address this, we propose and develop MKLDROID, a unified framework that systematically integrates all the aforementioned views in the hopes that, while a malware app can disguise itself in some views, disguising in every view while maintaining malicious intent will prove to be substantially more difficult. MKLDROID leverages on CWLK to place a structural & context-aware similarity metric on each distinct view and then employs Multiple Kernel Learning (MKL) to find a weighted combination of the views which yields the best classification accuracy with a Support Vector Machine (SVM) classifier. Beside integrating multiple views, MKLDROID's salient trait is its ability to localize fine-grained malice code portions (e.g., classes/methods) from app's PRGs. This trait not only help to assess the detection model's trustworthiness but also cater several important applications such as supporting human analysts studying malware behaviors, and engineering malware signatures.

3. We propose SUBGRAPH2VEC, a novel approach for learning latent representations of

rooted subgraphs from PRGs inspired by recent advancements in Deep Learning. SUBGRAPH2VEC leverages on local information obtained from PRG neighborhoods of nodes and subgraphs to learn their latent representations in an unsupervised fashion. We demonstrate that subgraph vectors learnt by our approach could be used in conjunction with classifiers such as Convolutional Neural Network, SVMs and relational data clustering algorithms. Also, we show that the subgraph vectors could be used for building a deep learning variant of Weisfeiler-Lehman graph kernel. Our experiments on several large-scale real-world datasets reveal that SUBGRAPH2VEC achieves significant improvements in accuracies over existing graph kernels (incl. CWLK) on both malware and clone detection tasks.

In sum, this thesis proposes two methods for learning representations of PRGs namely, CWLK and SUBGRAPH2VEC. With the PRG embeddings thus built, we address four specific issues that plague Android malware and clone detection approaches, namely, population drift, integrating multi-view features to achieve comprehensive detection, localization of malicious code portions and scalability, heavily leveraging on ML techniques such as Online Learning, Kernel Methods, MKL and Deep Learning.