OS Android Attack - Task 02

–

HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network. Weak spots and exploit them

Gilad Livshitz 206768962

Shay Peretz 203464870

December 21st 2022

Abstract

Android has become the most popular mobile platform, and a hot target for malware developers. At the same time, researchers have come up with numerous ways to deal with malware. Among them, machine learning based methods are quite effective in Android malware detection, the accuracy of which can be as high as 98%. Thus, malware developers have the incentive to develop more advanced malware to evade detection. This paper focuses on HinDroid Malware Detection System and presents the weak spots of the method and presents an adversary attack scenario (Collusion Attack) that will compromise this malware detection method. The malware developers can perform this attack easily by splitting malicious payload into two or more apps. Meanwhile, attackers may hide their malicious behavior by using advanced techniques (Evasion Attack), such as obfuscation, etc. According to research, 87.4

1 Introduction

Smartphones have been widely used in people's daily life, such as online banking, automated home control, and entertainment. Due to the mobility and ever-expanding capabilities, the use of smartphones has experienced an exponential growth rate in recent years. Android, as an open-source and customizable operating system for smartphones, is currently dominating the smartphone market. However, due to its large market share and open-source ecosystem of development, Android attracts not only developers for producing legitimate Android applications (apps), but also hackers to disseminate malware (malicious software) that deliberately fulfills the harmful intent to the smartphone users. Because of the lack of trustworthiness review methods, developers can upload their Android apps including repackaged apps, ransomware, or trojans to the market easily in even Google's official Android market. To protect legitimate users from the attacks of Android malware, currently, the major defense is mobile security products, such as Norton, Lookout and Comodo Mobile Security, which mainly use the signature-based method to recognize threats. However, attackers can easily use techniques, such as code obfuscation and repackaging, to evade detection. The increasing sophistication of Android malware calls for new defensive techniques that are robust and capable of protecting users against novel threats.

2 HinDroid Malware Detection System

2.1 HinDroid Method

To be more resilient against the Android malware's evasion tactics, instead of using Application Programming Interface (API) calls only, HinDroid analyzes the relationships among them, whether the extracted API calls belong to the same code block, or if they with the same package name, or use the same invoke method, etc. Relations between APIs and apps and different types of relations among apps themselves can introduce higher-level semantics and require more effort for attackers to evade detection.

The HinDroid model calculates similarities between apps and feeds the similarities to a SVM to form a decision boundary between two data classes. It constructs a Heterogeneous Information Network (HIN) to capture the relationships between apps and between APIs. The network consists of two node types and four edge types. Edge type A connects apps to APIs if the API is used by that app. Edge type B connects every pair of APIs that are co-appeared inside the same code block in every app. Edge type P connects every pair of APIs that are from the same library (package). Edge type I connects every pair of APIs that are using the same invoke method for present in every app. Each type of these edges will be represented by an adjacency matrix. To calculate the similarity, we formalize it as the number of common features that are both present in the HIN. If the similarity between two apps is high, there may be something to be said for it. We define this common feature as the number of metapaths between apps. A metapath starts from an app and ends with an app and it goes through a symmetric node path in the HIN. For example, A-A means how many APIs are common within a pair of apps; A-P-A means how many pairs of APIs (one API from ai, one API from aj) that use a common library. HinDroid assumes these metapaths represent a unique common feature between applications. For the case of metapath A-B-A, if two apps both called these specific APIs within a block somewhere in the source code, then their functionality or intent should be similar. Based on this intuition, we can see that if an app is more similar to a malware in terms of high similarity, we can be more confident that the mystery app should be a malware as well. In implementation, these meta paths are calculated using multiplication of adjacency matrices, and the result is used as a Gram matrix for the SVM algorithm to find a decision boundary between the two classes.

2.2 HinDroid Method Weak spots and how to use them

When we analyzed the HinDroid methods, we find out HinDroid analyzes the relationships among the API calls in a single APK file. That leads to the idea to split the malware attack into several files instead of one also called a Collusion attack. For using this attack we need to analyze the malicious behavior to figure out if it is possible to separate the operation into two or more sub-procedure. Also, its requires knowledge of the SVM classification model to can maximize the effect of a Collusion Attack.

Another weakness of the HinDdroid is counting the number of API calls, the number of libraries used, the number of code blocks used, and max number of code blocks within a file number of invokes for every type. The assumption is the malware files are smaller in every way. If we increase the number of blocks and libraries et cetera, the classifier can classify our files as unmalicious files.

3 CONCLUSION

In this paper, we show the HinDroid method. instead of using API calls only for feature representation, it further analyzes the relationships among them, which create higher-level semantics and requires more effort for attackers to evade the detection. We present the weeks that we found, considering only one file analysis and using features that attackers can manipulate.