Android Attacks: An Attack on an FM Drebin Classifier

Amit Waizman, Rivka Buskila

January 18, 2023

Abstract

Recently, an effort has been made to use machine learning models and methods to characterize and generalize malicious behavior patterns of mobile apps for malware detection. We used an Android malware detection classifier based on a disassembly machine architecture and extracting features of an Android application from manifest files and source code. This method achieved a test result of 100.00% accuracy in the DREBIN dataset and an accuracy score of 99.22% with a false positive rate of only 1.10% in the AMD data set [1]. In this paper, we propose an attack method that can overcome this classifier and successfully classify malicious apps as good by adding "positive noise" to a malicious app while maintaining its functionality and no damage.

1 Introduction

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for mobile devices with touch screens, such as smartphones and tablets. More than 70% of smartphones based on the Android open source project run the Google ecosystem.

Android has been the best-selling operating system in the world on smartphones since 2011 and on tablets since 2013. As of May 2021, it had over three billion monthly active users, the largest installed base of any operating system, and as of January 2021, the Google Play Store featured over 3 million apps. To protect mobile phone users, several antivirus companies, such as McAfee and Symantec, provide software products as a central defense against these types of threat, these products typically use a signature basis A method [5]
to identify threats.Signature-based methods including creating a unique signature for everyone known in the past. Android malware detection using classified in Drebin style based on FM Patterns in previously undetected malware samples. One. The main type of malware detection is based on machine learning. The method is called static analysis [1], [6] In the framework of this article, we would like to attack the classifier and damage its credibility.

In the application area there is a difficulty in adding "noise",Since any action can have a negative effect such as harming the functionality of the application, so apart from the attempts to evade the classifier, it should be noted that the changes we have made do not harm the functionality of the application. In order to carry out the attack with this method, we add "positive noise" in malicious applications, This is by adding key changes to the manifest file and additional changes to the

smali file by add directorys
The model was trained using the dataset consisting of 80,000 apps, of which 6,000 are malicious. Furthermore, we intend to analyze our results by testing them against a high reliability classifier to see if we have been able to compromise the reliability of this classifier.

2 Related Work

Android malware is a new but rapidly growing threat. Classic defenses, such as antivirus scanners, are increasingly failing to cope with the amount and variety of malware in the apps markets. Although recent approaches, support the filtering of such applications Outside of these markets, they incur runtime overhead, that is, prohibit direct protection of smartphones. as a medicine, This paper [1] presents DREBIN, a lightweight detection method of Android malware. DREBIN combines concepts from static analytic and machine learning, allowing it to improve the rate of malware development.
The factorization machine (FM) [3], which is a general predictor like SVMs, is also able to estimate very reliable parameters, high sparsity. The models of the dismantling machines are all nested Variable interactions (compared to the polynomial kernel b SVM), but uses a factorized parameterization instead of a Dense parameterization as in SVMs. We show that the FMs model equation can be calculated in linear time and that it depends only on a linear number of parameters. It allows direct optimization and storage of model parameters without the need to store any training data (eg support vectors) for prophecy. In contrast, non-linear SVMs are usually double adjusted, and predictive computer (e the model equation) depends on parts of the training data (e supports vectors).
The study [2] presented some innovative evasion attacks against Popular Android malware detection systems, such as Drebin, Sec-SVM, FM and DNN. Evasion attacks included the Masking of permission requests . The recognition rate of the systems decreased, ranging from 13in the case of Sec-SVM to almost 97% in the case of DNN.
In this article, we will implement a type of attribute-based attack, which includes adding values to manifest file and Smali code files.

3 Methodology

For this study we used a gray box attack We used limited information to identify the strengths and weaknesses of the classifier This is by studying the structure of the input and output obtained from the classifier after running it while referring to the features that characterize the model We used a Drebin-style classifier based on FM [3] In addition, we used a data set containing about 74.000 normal applications and about 6000 malicious ones. Taken from [2] The ratio between benign and malicious apps is 90% benign and 10% malware In the published article [4] it was stated that this is also the real ratio in the world between benign and malware apps

The feature set used by the classifier: The classifier includes 7 types of features from the source code and the manifest file, of which four types are extracted from the manifest file of an application and three additional types of features from the source code of the unpacked application.

Attributes from a manifest file Application components: The components declared in the manifest file define the various interfaces that exist between the application and the end user, as well as between the application and the larger Android operating system as a whole. The names of these components are collected

to help identify versions of known malware.

: hardware features If an app wants to request access to the device's hardware components, such as its camera, GPS, or sensors, these features must be declared in the manifest file. Requesting certain hardware components or pairs of components may have security implications.

: Permissions Android uses a permission mechanism to protect user privacy. An application must request permission to access sensitive data (e.g. SMS), system features (e.g. camera), and restricted APIs. Malware tends to ask for a certain set of permissions. In this sense, it is similar to how we handle hardware properties.

: Intent filter Intent filters declared as part of the declaration of components in the manifest file are important tools for communication between components and between applications. Intent filters define a special entry point for both the component and the application. Intent filters can be used to intercept specific intents. Malware is sensitive to a special set of system events. Thus, intent filters can serve as essential features.

Features of the source code:

: Limited APIs In the Android system, some special APIs related to accessing sensitive data are protected by permissions. If an application calls these APIs without requesting appropriate permissions, this may be a sign of a root exploit.

: Suspicious APIs We should be aware of a special set of APIs that can lead to malicious behavior without asking for permissions. For example, cryptography functions in the Java library and some math functions do not need permission to be used. However, these functions can be used by malware for code obfuscation. Therefore, it should be noted that there is an unusual use of these functions. We mark these types of functions as suspect APIs.

Usage permissions: We first extract all API calls from the application's source code, and use this to build a set of permissions actually used in the application by looking up a predefined dictionary that associates an API with its required permissions. The classifier type that you are statically parsing.

Our attack method has several main steps: in first step, We took an randomly apps from the goodware test and sent them to the classifier to verify that he classifies them as benign. We examine their structure and how we can use this structure to evade the classifier in malicious applications. Specific changes in the manifest file which is a central part of the application structure: We found shared permission: In the second step, we noticed that

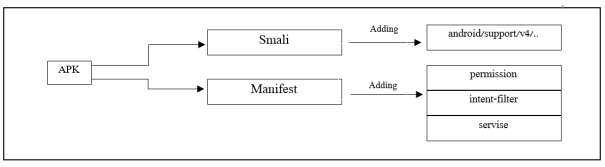
```
<uses-permission android:name="com.Aapp.UlagaTamilOli.permission.C2D_MESSAGE"/>
```

most of the benign apps contain the following service and intent-filter: We added these fields

```
<service android:name="com.arellomobile.android.push.PushGCMIntentService"/>
 <intent-filter>
       <action android:name="com.google.android.c2dm.intent.REGISTRATION"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.AlertDialogs"/>
    </intent-filter>
```

as "noise" to all malwares. In the third step we referred to the source code for a smali file and saw that most of the best applications in the test contain the V4 libraries. V4 contains additional libraries, and we added this folder to the rest of the malware applications: **android/support/v4** The change process can be seen in Figure 1

Figure 1: Diagram of the change process in the application.



4 Results

We offer a solution to reduce the classified accuracy of drebin type based on FM.
For the purpose of the proposed solution, it included changes in the manifest file and the smali file.
Below are the results of the classification according to and after the change. In the results, you can see the execution of malicious applications. At this stage we only ran the classifier on tests of malicious apps to see if the evasion was successful

Malware	Accuracy	recall	f1-score
Before	0.97	0.98	0.99
After	0.72	0.73	0.84

From the results, we can see that we managed to escape the Drebin classifier. Before the change, the classifier was able to identify 97% malicious apps, indicating high and accurate reliability. Well, after the attack, we see that the changes we made in the manifest file and the small file led to damage to the reliability of the classifier and lowered Amino by about 20

It can be seen that by adding "noises" to a malicious application, it contributes to evading the classification while maintaining the functionality and without harming the original application. Causes a higher similarity between a good application and a bad one in test areas.

5 Conclusion

In this work, we proposed the ability to create malware attacks against FM-based drebin classifier by implementing attacks on the data set of malicious applications from the test applications, The attack was carried out by adding

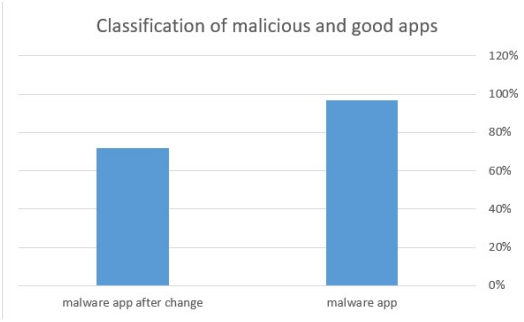


Figure 2: Before change Accuracy=0.97 and after change Accuracy=0.72

"noise" to a malicious application in order to bring the imagination between a good application and a malicious one closer Evasions are a promising strategy for improving an opponent's attacks Real malware. Our test results show that we can use this information to evade the classifier without making the malicious application defective. As future work, our aim is to improve the efficiency of the system by comparing different types of features and checking which one brings the highest evasion. In addition, we would try to explore other directions of changes in the structure of the application.

References

[1] D. Arp et al. "DREBIN: Effective and explainable detection of Android malware in your pocket". In: *Proc. NDSS, vol. 14* (2014).

[2] Harel Berger et al. "Crystal Ball: From Innovative Attacks to Attack Effectiveness Classifier". In: *IEEE Access* (2021).

[3] CHENGLIN LI et al. "Android Malware Detection Based on Factorization Machine". In: *IEEE Access* (2019).

[4] F. Pendlebury et al. "TESSERACT: Eliminating experimental bias in malware classification across space and time". In: *Proc. 28th USENIX Secur. Symp.* (2019).

[5] D. Venugopal and G. Hu. “Efficient signature based malware detection on mobile devices”. In: (2008).

[6] D.-J. Wu et al. “Droidmat:Android malware detection through manifest and API calls tracing”. In: *Proc. 7th Asia Joint Conf. Inf. Secur* (2012).