MalDetector: A Permission and Inter-Component Communication-Based Detection Approach

REU fellow: Evelyn Bailey[1]

Faculty Mentor: Dr. Wenjia Li[2]

Affiliation: [1.] University of Pennsylvania, [2.] School of Engineering and Computing Science, NYIT

Emails: ebail@seas.upenn.edu, wli20@nyit.edu

ABSTRACT

Despite efforts to detect malware using permissions listed in the manifest of apps, new forms of malware use Android's inter-component communication (ICC) system to bypass permissions. We develop a detection framework for statically analyzing Android apps using a hierarchical approach that analyzes permissions and then uses information gathered from Android's ICC to achieve higher accuracy. Our system, MalDetector, is tested on a data set of 745 good apps from the Google Play Store and 745 bad apps from the Drebin [1] dataset. We implemented it using three machine learning algorithms, and MalDetector achieved a highest precision of 95.1% using a Neural Network.

INTRODUCTION

Despite Android's immense popularity, it accounts for over 80% of malware attacks [2]. Just as researchers have been able to detect malware using permissions, malware attacks have also been evolving to bypass permissions and exploit Android's ICC system. Recent research has begun using machine learning to classify and detect deviant behavior using ICC information. Some approaches, like RoppDroid [3], seek to modify Android's ICC so that it reduces the permission set of the apps carrying out the function (deputies) to that of the apps making API Calls (senders). Others, like ICCDetector [4], use ICC information to create feature vectors and process them through a Support Vector Machine. We seek to combine the advantages of detecting malicious apps using risky permissions with detecting malicious apps using Android's ICC.

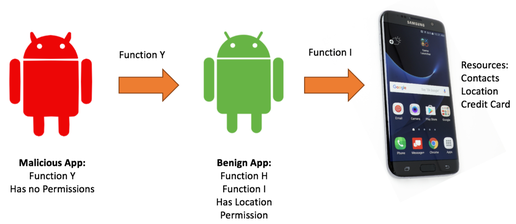


Fig. 1: Android's ICC System

METHOD

Information was extracted, decoded, and processed using:

APKTOOL, scikit learn, IC3[5]

1) MalDetector first extracts the set of permissions each app requests from the manifest of each app using APKTool.

2) It then uses one of three ML algorithms (Support Vector Machine [SVM], Boosted Tree [BT], or Neural Network [NN]) to classify the data set using permissions.

3) MalDetector extracts the set of intents each 'good' app calls and any Services, Providers, or Receivers it includes in the manifest using IC3Tool.

4) Finally, MalDetector uses the same ML algorithm from Step 2 to classify the remaining apps and outputs the final classification of good and bad apps.

SYSTEM FRAMEWORK

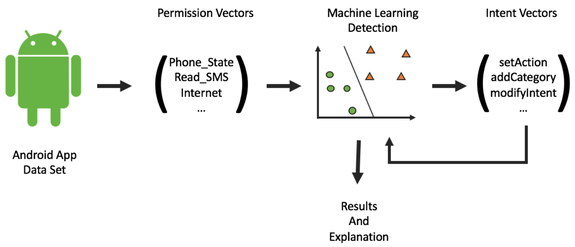


Fig. 2: Project Framework

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
<permission android:name="edu.upenn.cis350.g8.when2meetformobile.permission.C2D_MESSAGE"
android:protectionLevel="signature"/>
<uses-permission android:name="edu.upenn.cis350.g8.when2meetformobile.permission.C2D_MESSAGE"
/>
<application android:allowBackup="true" android:debuggable="true" android:icon="
@mipmap/ic_launcher" android:label="@string/app_name" android:roundIcon="
@mipmap/ic_launcher_round" android:supportsRtl="true" android:theme="@style/AppTheme">
    <activity android:name="edu.upenn.cis350.g8.when2meetformobile.MainActivity" android:
    screenOrientation="portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
```

Fig. 3: Example Manifest File

Feature Space	Information
Permissions Only	Analyze the list of all permissions requested in the manifest.
ICC Only	Analyze the list of all Intents, Providers, Services, or Receivers in the manifest.
Permissions and ICC	Analyze the list of all permissions and ICC information at the same time.
Hierarchical Approach	Analyze the permissions first and then analyze based on ICC information only.

Fig. 4: Feature Spaces

In addition to the hierarchical approach, we also implemented three additional approaches for comparison. A feature space is defined based on what information we include from the manifest as vectors for each machine learning algorithm.

RESULTS

Fig. 5: Precision Results for Each Feature Space Using SVM, Boosted Trees, and Neural Networks

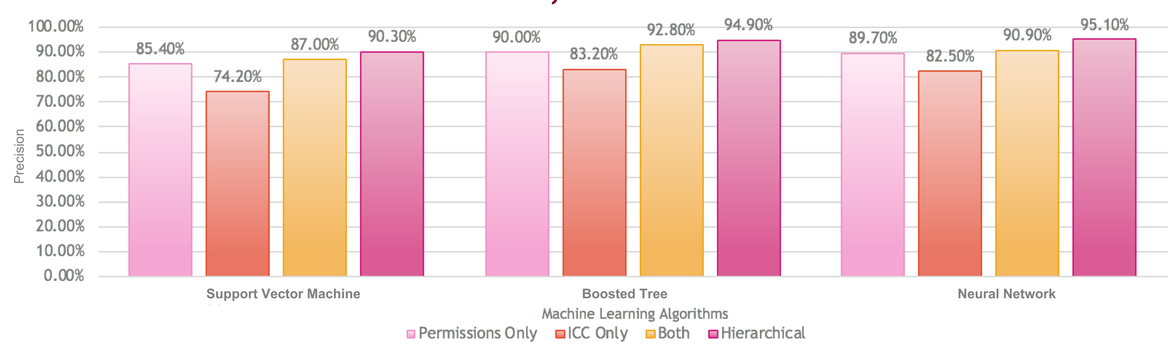
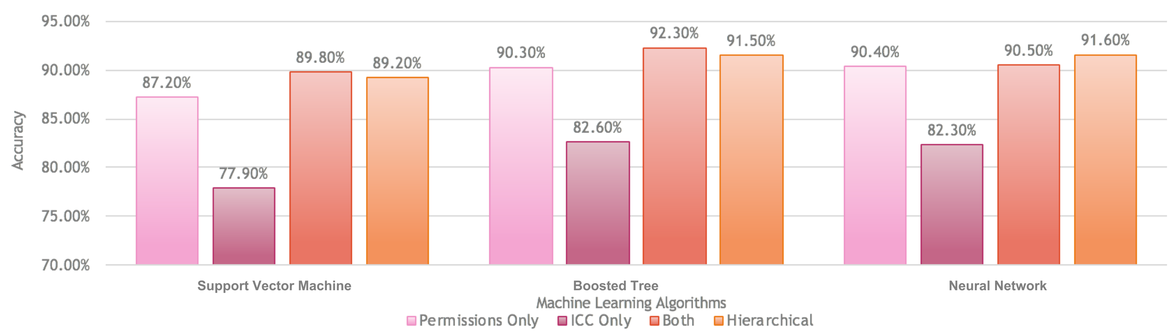


Fig. 6: Accuracy for Each Feature Space Using SVM, Boosted Trees, and Neural Networks



DISCUSSIONS

In comparison to the other feature spaces, the hierarchical approach outperformed the Permissions-Only and ICC-Only feature spaces and performed similarly to the Both-Permissions-and-ICC feature space in accuracy. The hierarchical approach was able to reduce the number of false positives relative to the other approaches, and it is noted that it could be because the ICC analysis was better at correctly classifying malware than classifying good apps. The similarity in results between the highest three performing feature sets could be due to the age of the Drebin dataset ('14), which may have older malicious apps that rely more on permissions than ICC and may not be representative of current malware trends.

CONCLUSIONS

We have investigated Android malware detection using a hierarchical approach based on permissions and ICC components. We successfully built a framework that extracts the permissions and ICC components from a dataset of 1,490 apk files, half of which are good and half of which are bad. We implemented it using three machine learning algorithms and three additional feature spaces for comparison. The hierarchical approach achieved the highest precision of 95.1% using Neural Networks, and the highest accuracy of 92.3% was achieved using the Both-Permissions-and-ICC feature space and Boosted Trees.

FUTURE WORK

- Implement a permission-based malware detection algorithm that has a lower false positive rate.
- Find a feature selection method that reduces the feature space but improves accuracy.
- Investigate more distinguishing features related to ICC than Intents, Services, Providers, and Receivers.

REFERENCES

[1] Arp, Daniel, et al. "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket." *Ndss*. Vol. 14. 2014.

[2] Internet Security Threat Report, Vol 23. March 2018.

[3] Dai, Ting, et al. "Roppdroid: Robust permission re-delegation prevention in android inter-component communication." *Computers & Security* 68 (2017): 98-111

[4] Xu, Ke, Yingjiu Li, and Robert H. Deng. "ICCDetector: ICC-based malware detection on Android." *IEEE Transactions on Information Forensics and Security* 11.6 (2016): 1252-1264.

[5] Octeau, Damien, et al. "Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis." *Effective Inter-Component Communication Mapping in Android with Epicc: An Essential Step Towards Holistic Security Analysis* (2013).

[6] Li, Wenjia, Jigang Ge, and Guqian Dai. "Detecting malware for android platform: An svm-based approach." *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*. IEEE, 2015.

ACKNOWLEDGEMENT

This research was made possible by the National Science Foundation (NSF) Research Experiences for Undergraduates (REU) program. I would like to thank all of the mentors and research fellows at the New York Institute of Technology who have provided insights and assistance while conducting this research. I also want to sincerely thank Dr. Wenjia Li for his guidance, support, and encouragement throughout this research experience.

This project is funded by National Science Foundation Grant No. CNS-1559652 and New York Institute of Technology.