Constraints on Permissions for Making Safe App Execution

Supervised By:
Prof. R.K. Shyamasundar

Presented By:
Dharmendra Kumar

Overview

- Is data and resources on your smartphone secure?
- How can we secure our data from malicious application?
- Does current security schemes provide required level of security?
- How does current security scheme fail?
- How to protect an application from changing its behaviour?

Problem

- Using Dynamic and Static analysis extracts various informations from applications as traces.
- Build a classification model to check whether applications are benign or malicious based on traces.
- Stop the malicious intent of applications using wrapper around them.

Motivation

- Malicious applications frequently easily passes the security suite of Google’s Play Pro-tect security suite, and some applications attracts millions of downloads before Google can find and remove them.
- Recently the security firm Check Point discovered a new typeof Android malware called “Expensive Wall” hidden in 50 apps in the Play Store.
- They have been cumulatively downloaded nearly 4.2 million times.

Motivation (contd.)

- Even after Google removed all these applications, Check Point discovered new sample of the malware in the Google Play store.
- So, we can not rely only on Google Play Protected security suite.

Introduction

- Android mobile devices is becoming very popular due to ease of use and its user-friendliness User Interface(UI).
- Nearly 82% of smartphones run Android Operating System.
- Due to such a large market share, malware authors are targeting Android mobile devices.
- Android tops 2016 vulnerability list with 523 Common Vulnerabilities and Exposures(CVE).

Introduction (contd.)

- Following are some data about vulnerabilities in different software/OS:
  - Debian Linux(Debian) - 319
  - Debian Linux(Ubuntu) - 278
  - Mac OSX (Apple) - 215
  - Windows 10 - 172
  - Iphone OS - 161
- In 2016, Kaspersky Lab detected the following:
  - 8,526,221 malicious installation packages
  - 128,886 mobile banking Trojans
  - 261,214 mobile ransomware Trojans

Introduction (contd.)

- According to AO Kaspersky Lab, following Android security issues were in trend in 2016:
  - Malicious program using super-user rights.
  - Emergence of new ways to bypass Android protection mechanism.
  - Mobile ransomware.
  - Active development of Mobile Banking Trojans.
  - Infecting mobiles by putting malicious application in Google Play.

Previous Approach and Techniques

- Permission based approach
  - To stop malware installation permission based system has been developed. While installation applications ask for permission to access sensitive resources.
  - Based on requested permission a classifier has been developed to identify benign and malicious application.
- Two-layered permission based approach
  - In this method, they analyzed data from two types of permissions namely requested permissions and used permissions.
  - After analyzing requested permissions, they make classifier to classify applications int benign and malware.
  - After this classification, they make another classifier by analyzing used permissions to identify malware.

Previous Approach and Techniques (contd.)

- Malware detection using static feature-based analysis
  - This approach uses various static information such as permissions, deployment of components, intent message passing and API calls for characterizing the Android application behaviour.
  - After extracting the above information one can either applies K-means algorithm to classify applications into benign and malware or generate a model using KNN classifier for detecting malware.
- Malware detection using hybrid of static and dynamic analysis
  - By static analysis, it extracts expected activity switch paths by analyzing both Activity and Function Call Graphs.
  - Then it uses dynamic analysis to traverse each UI elements and explore the UI interactions paths towards sensitive APIs.
  - Using these data, they create a model to detect malware.

Previous Approach and Techniques (contd.)

- Behaviour-based malware detection system
  - They firstly gather test data and actual processed data, it processes the data and try to figure out some patterns in the data that can be used to differentiate benign applications and malicious applications.
  - They use different machine learning techniques to find the pattern in the data.
  - After finding the pattern, they make classifier to identify the malware.
- By analyzing usage of sensitive data
  - In this method, they firstly analyze the flow of sensitive data in benign applications and then try same for malicious applications.
  - From these data, they implements different type of classifier to detect malicious applications.

Problems with Current Security Scheme

- Android current security is based on permissions.
- App needs all permissions to work properly.
- If restrict any permissions, app does not work properly.
- Apart from permissions, there are more things that can be used to exploit the private data of users e.g clipboard, view.
- Clipboard is buffer which is used to hold the data copied from any application.
- Currently, there is no mechanism to stop applications from reading data from clipboard.

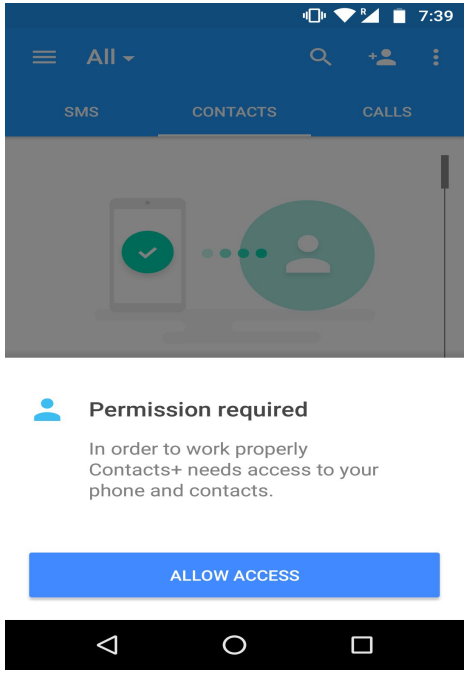

Fig: After restricting contact

Problems with Current Security Scheme (contd.)

- Suppose an application needs permission for sending messages only, so the application will ask accessing messages.
- In current scenario, application will be given access to the message.
- After some time application will change its behaviour and starts reading messages also because it has access to messages.
- After reading sms, application can send sensitive information to third party. Sensitive information in this case can be OTP, personal messages, messages from bank, etc.

Our Approach

- Our approach has three phases:
  - Analysis of applications.
  - Determining the behaviour of applications.
  - Restricting the malicious behaviour of applications.
- We have used DroidMate and RiskInDroid for dynamic analysis and static analysis respectively.
- We have used AndroPyTool for both static and dynamic analysis.
- We have extracted permissions and traced API calls for each applications.

Our Approach

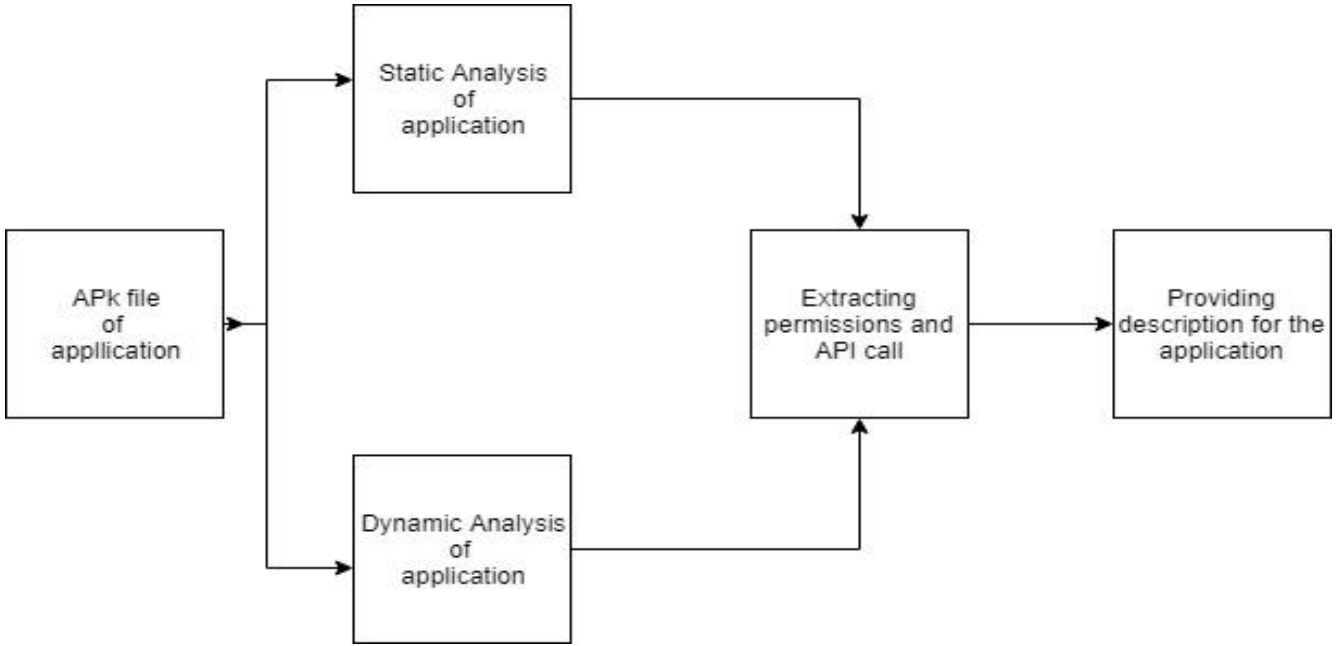

Figure: Workflow of Analysis Phase

Our Approach(contd.)

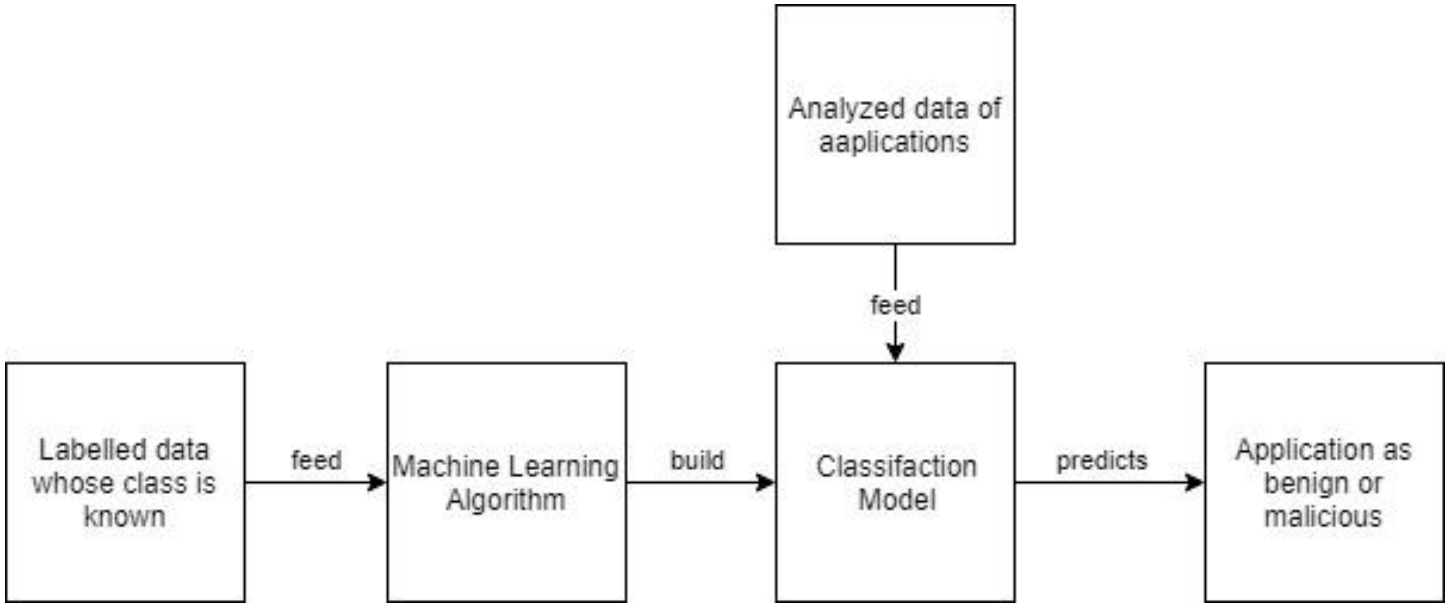

Figure: Workflow of Classification Phase

Our Approach(contd.)

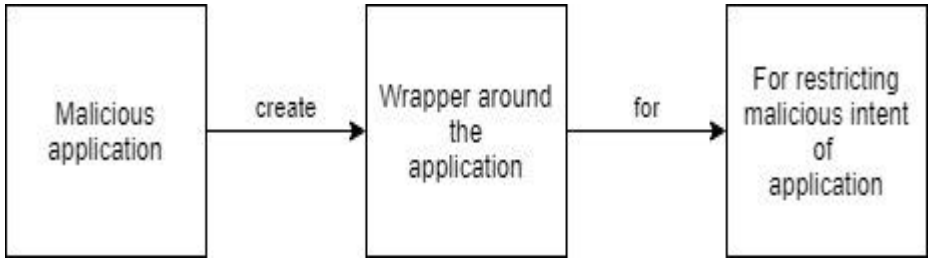

Figure: Workflow of Restriction Phase

Our Approach (contd.)

- We have classified the apps as benign or malicious using following machine learning techniques:
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Neural Network
  - Random Forest
- If applications are classified as malicious, we have restricted its malicious behaviour by creating a wrapper around it.

Analysis of Applications

- We have done the analysis of following applications:
  - Budget Planner
  - BHIM
  - Funnyys
  - Omingo
  - System Certificate
  - Tez
  - MMS Beline
  - Google Calendar
  - Laughter
  - Android Framework

Analysis of Applications (contd.)

- We have extracted the following types of permissions from the applications:
  - Declared permissions
  - Requested and used permissions
  - Requested but not used permissions
  - Not requested but used permissions
- We have also traced the API calls and event which is calling that API.

Analysis of Applications (contd.)

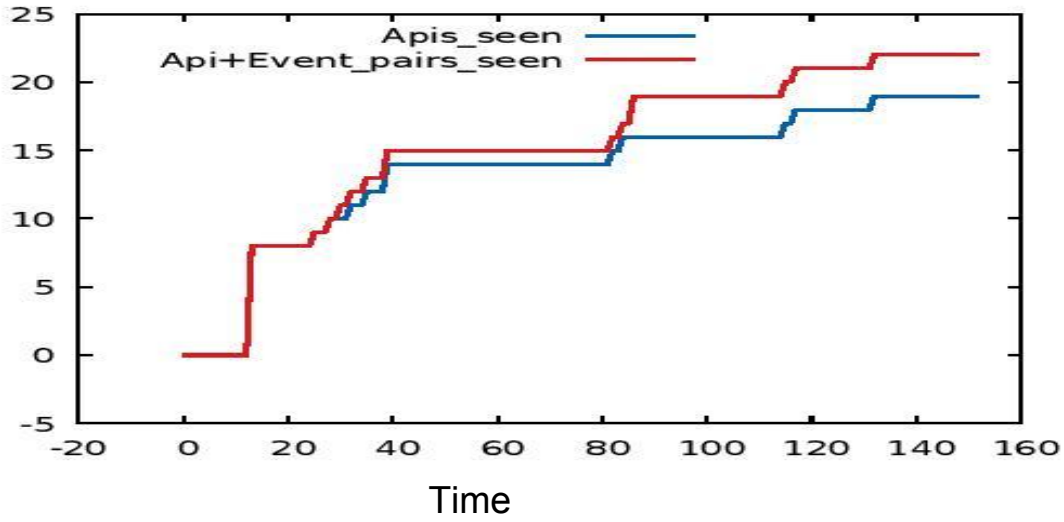

Fig: Number of Unique API calls seen during analysis of Google Calendar.

Analysis of Applications (contd.)

- Following are the list of permissions found during analysis of BHIM app:
  - android.permission.SEND_SMS
  - android.permission.RECEIVE_SMS
  - android.permission.READ_CONTACTS
  - android.permission.CALL_PHONE
  - android.permission.READ_SMS
  - android.permission.READ_PROFILE
  - android.permission.RECEIVE_BOOT_COMPLETED
  - android.permission.WRITE_EXTERNAL_STORAGE
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.ACCESS_COARSE_LOCATION
  - android.permission.INTERNET
  - android.permission.ACCESS_NETWORK_STATE

Analysis of Applications (contd.)

- in.org.npci.upiapp.permission.C2D_MESSAGE
- android.permission.WAKE_LOCK
- android.permission.ACCESS_WIFI_STATE
- android.permission.CAMERA
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.READ_PHONE_STATE
- com.google.android.c2dm.permission.RECEIVE
- android.permission.VIBRATE

Analysis of Applications (contd.)

- Following is the screenshot taken during analysis of Omingo application:

Classification Model

- Logistic Regression
  - Model classifies 93.8% of benign correctly and 83.4% of malicious app correctly.

Actual	Predicted	
	Benign	Malicious
Benign	93.8	6.2
Malicious	16.6	83.4

- Benign apps: Budget Planner, BHIM, System Certificat, Tez and Google Calendar.
- Malicious apps: Funnyys, Omingo, MMS Beline, Laughtter, Android FrameWork

Classification Model (contd.)

- Support Vector Machine (SVM)
  - Model classifies 92.6% of benign correctly and 78.7% of malicious app correctly.

Actual	Predicted	
	Benign	Malicious
Benign	92.6	7.4
Malicious	21.3	78.7

- Benign apps: Budget Planner, BHIM, Funnyys, System Certificat, Tez and Google Calendar.
- Malicious apps: Omingo, MMS Beline, Laughtter, Android FrameWork

Classification Model (contd.)

- Neural Network
  - Model classifies 96.3% of benign correctly and 86.6% of malicious app correctly.

Actual	Predicted	
	Benign	Malicious
Benign	96.3	3.7
Malicious	13.4	86.6

- Benign apps: Budget Planner, BHIM, Tez and Calendar.
- Malicious apps: Funnyys, Ominog, System Certificate, MMS Beline, Laughtter, Android FrameWork

Classification Model (contd.)

- Random Forest
  - Model classifies 97.2% of benign correctly and 87.1% of malicious app correctly.

Actual	Predicted	
	Benign	Malicious
Benign	97.2	2.8
Malicious	12.9	87.1

- Benign apps: Budget Planner, BHIM, Tez and Calendar.
- Malicious apps: Funnyys, Ominog, System Certificate, MMS Beline, Laughtter, Android FrameWork

Classification Model (contd.)

- Comparison

Algorithms	Benign	Malicious
Logistic Regression	93.8	83.4
Support Vector Machine	92.6	78.7
Neural Network	96.3	86.6
Random Forest	97.2	87.1

Restricting Malicious Intent of Applications

- We can restrict malicious behaviour mainly two ways:
  - By modifying application.
  - By without modifying application.
- Modifying application is tedious task, we can not modify if app is encrypted.
- We have used Xposed framework.
- Instead of modifying corresponding DEX or ART representation of the application, it is deeply integrated with the Android system where it replaces some system components after backing up the originals.

Restricting Malicious Intent of Applications (contd.)

- There are multiple advantages to this approach. Some of them are:
  - you can modify parts of the system you otherwise could not.
  - you can apply multiple modifications to the same app in a combination best fitting your intentions.
  - changes are easy to undo: As all changes are done in the memory, you just need to deactivate the module and reboot to get your original system back.
- Zygote is the main process of the Android runtime. Every application is started as a copy (“fork”) of it.

Restricting Malicious Intent of Applications (contd.)

- This process is started by an */init.rc* script when the phone is booted.
- The process start is done with */system/bin/app_process*, which loads the needed classes and invokes the initialization methods.
- Xposed framework replaces the existing *app_process* with *extended app_process* after taking backup of it.
- This extended startup process adds an additional jar to the classpath and calls methods from there at certain places.

Restricting Malicious Intent of Applications (contd.)

- We have modified the modules of framework to restrict the malicious behaviour of applications.
- When an application calls any method, we can do following three things with that method:
  - Leave it unmodified.
  - Change the parameter.
  - Modify the body part.

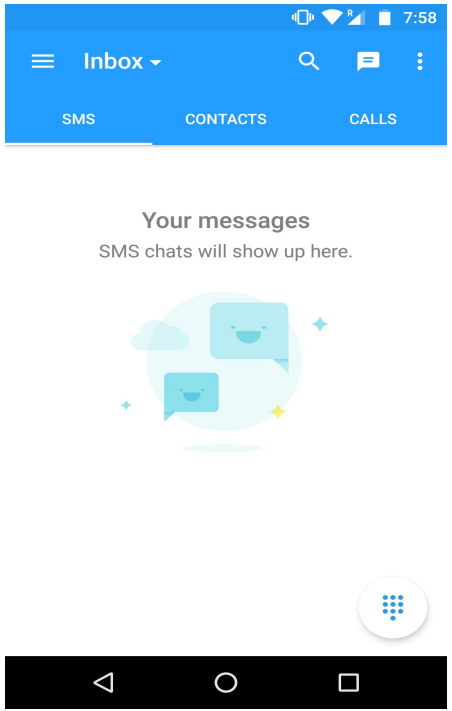

Fig: After restricting message

Restricting Malicious Intent of Applications (contd.)

- If application asks to read contact, we can either give no data or fake data.
- Now we can also restrict the permission to access data from clipboard.
- We can also provide fake sim number, IMEI number, device id, etc.

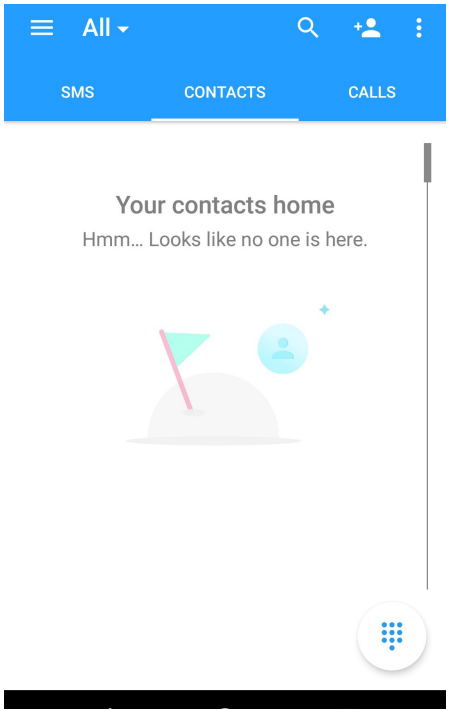

Fig: After restricting contacts

Conclusion

- We have successfully classified the applications into benign and malicious.
- Following applications are classified as benign:
  - Budget Planner
  - BHIM
  - Tez
  - Google Calendar
- We have not restrict any permissions for benign applications.

Conclusion (contd.)

- Following applications are classified as malicious:
  - Funnyys
  - Omingo
  - System Certificate
  - MMS Beline
  - Laughter
  - Android Framework
- We have restricted the permissions which can cause the malicious behaviour.

Thank You