ISSN:0975-9646
Gunjan Kapse et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (4) , 2015, 3561-3564

Detection of Malware on Android based on Application Features

Mrs. **Gunjan Kapse**
Computer Science Department
JSCOE
Pune, India

Prof. **Aruna Gupta**
Information Technology Department
JSCOE
Pune, India

Abstract— **Threat of mobile malware is increasing day by day. Since Android is the most popular and maximum sold mobile phone, there is an increasing threat of malware on Android based mobile device. The different antimalware products available in market can detect the malware in its original form. But they cannot detect the malware after applying some form of obfuscation or transformation to the malware. The malware detection method based on permission feature of application can lead to many undetected malwares. This paper proposes to do more comprehensive static analysis of application covering more features, in addition to permission features. This increases the malware detection strength. The different features which would be analyzed are permissions and suspicious API calls. Doing this would detect the malware with more accuracy. The application would be classified as benign or malware correctly. First the permissions are extracted from the manifest file and the API calls are extracted from disassembled code. Weights are assigned to permissions and API calls based on their malicious nature. If the total weight of permissions and API calls of an application exceed a predefined threshold, then the application is categorized as malware.**

Keywords— Malware; Mobile; Android.

I. INTRODUCTION

The number of android applications infected with malware in Google play store has become almost 3 to 4 times in 2014 as compared to 2012. Cyber criminals use malware to steal personal information of user. They can access credit card details; can give ads of particular product when a person visits a particular web site. Since the use of Android smart phones is increasing day by day, the malware is spreading at a very fast rate. The malware gets into mobile by downloading applications like malicious games, malicious software, and songs from different websites. Some times while installing some software, it asks for various permissions like Send SMS, internet access, root permission etc. If user gives these permissions in hurry without reading them completely, then the malware can take over these permissions and do its bad work.

Malware detection technique which is permission based, is not accurate method of detecting malware [1, 2]. Since same permissions are used by good ware and malware applications, only permissions system is not sufficient to categorize an application as benign or malware.

In this paper, we present more comprehensive static analysis approach of application. We use suspicious API calls, Filtered Intents and permissions to analyze and classify an application as benign or malware. First extract requested permissions and Filtered intents from Android manifest xml. Then Extract API calls from disassembled code. Disassembled code is obtained by Dex to jar conversion [3].

Comparison with Signature Based malware detection:

Most of the anti malwares in industry are signature based. The signature has to be added for every new malware coming up. Then antimalware is released with new updates. User has to update the Anti malware again and again. It slows down the device since large db of signatures has to be maintained.

Disadvantage of Signature based: 1. Update the antimalware again and again.2. High Space complexity: Large database of malware signature.

We have implemented behavior based detection which overcomes these disadvantages by not maintain the signature but analyzing the application features.

The different approaches of malware application analysis are:

Using static analysis**:** Statically analyze the application and checking its features like check different permissions, application components, different interfaces to components like broadcast receivers, services, activities etc. Many variants of a malware can have same service name. It performs the static program analysis that is not at run time. It analyses the logical structure, flow and data of the code. Checking API calls to for the privileges of application. Tool available for static analysis of application is Smali [4]. It is an open source tool for static analysis.

Using dynamic analysis: Analyze the application at runtime to see the malicious behavior. It involves dynamically monitoring the application in protected environment. But they cannot be deployed on phones and detect malicious software directly. They are not deployed on phones directly since they can damage and destroy all information on the machine. So they are implemented in controlled environment. It scans and analyses large volume of applications. It checks the actions of malware binary at runtime. It is proven better than static analysis. Hence this method is applied for offline detection.

II. LITERATURE SURVEY

Mihai Christodorescu, Somesh Jha [5] presented technique Semantic aware malware detection. It is based on semantic characteristics of program. Those properties are data flow and control flow. The binary program is disassembled first. Control flow graphs are created for each function of program, and then IR (intermediate representation) is generated using instruction transformer. Templates are created for comparison with malicious program. The

instructions in template are mapped with the instructions in program based on some rules. They both match if there is an assignment to variables of template node which matches the program node expression. It checks that the template variables should have similar update pattern as of program expression, although they may not have the same values.

Advantages:

1. It can handle different transformations/obfuscations like reordering of code or instructions. Register renaming can also be detected since it is doesn't check the names of variables in template instructions but it is based on matching function. Garbage insertion can be detected. 2. It has low value of false positives indicating that the tendency of detecting the good ware program as malware is very less.

Disadvantages:

1. It requires all the IR instructions in template to be of same form as that in program. For example, if the template has multiplication operation in the instruction and program has equivalent left shift operation, then they are not matched.
2. It enforce that the ordering of memory updates of program should be same as that of template.

Yu Feng, Saswat Anand [6] presented Apposcopy: Semantics-Based Detection of Android Malware through static Analysis. In first step it constructs ICCG (Inter component call graph) of application which shows different components, broadcast receivers, activities and services of application. In second step, it does static taint analysis that is it tracks the flow of data from source to sink.

Advantages: 1. High accuracy. 2. Can detect obfuscations like: Change of component, method, field names. 3. Method invocations to android classes are redirected using proxy methods.

Disadvantages:

1. Cannot detect unknown malware family.
2. Cannot detect obfuscations:
   a. Dynamic code loading
   b. Reflection + change of method or class name.
3. Cannot detect malware instantly.

Vaibhav Rastogi, Yan Chen, and Xuxian Jiang [7] proposed Catch Me If You Can: Evaluating Android Anti-Malware against Transformation Attacks. He introduced a framework which transforms the malware in different forms and test the transformed malware with commercial antimalware tools. The different obfuscations are:

A. Trivial Obfuscations

Those are the obfuscations which do not need code level changes. Following are the transformations in this category.

1) Repack: Unzip the apk (android application package), insert malware or malicious code and then repackage. After repackaging/zipping, the apk is released into the app store. And then zip again with different signature. The application is signed with custom keys.

2) Disassemble and reassemble: Disassembling and reassembling of apk is done by Apktool [12]. The jar is signed with some others name and again the new apk is created.

3) Change of package name: Package name of application is changed after getting the code. The code can be obtained using dex2jar [3].

B. DSA obfuscations

1) *Rename identifier*: Hackers after getting the APK of application run Baksmali [11] which is a disassembler. Hacker can get the byte code. Now hacker can rename method or field identifier and build it again to create new apk.

It is the category of obfuscation which can be detected by static analysis. Following are the obfuscations in this category.

2) Data encoding: The dex file contains all the strings and array data which is used in code. They can be used to develop signature against malware. These strings and array data are retrieved by hackers after applying reverse engineering by Apktool [12], encoded in some form and rebuilt. Thus they cannot be detected.

Daniel Arp, Michael Spreitzenbarth [8] gave method DREBIN which is effective and explainable detection of Android malware. This method enables identifying malware on smart phones directly. It combines static analysis of all the features of application and then apply machine learning to classify the applications as benign or malware. It also provides explanation for malware detection.

Advantages: It gives very few false alarms. The accuracy of method is high. It is good for detecting unknown malware.

Disadvantages: Cannot detect transformations that are not detectable by static analysis. For example Reflection and byte code encryption.

III. PROBLEM DEFINITION

To classify an application as benign or malware on Android mobile.

In Permission based Android Malware Detection technique given by Zarni Aung, Win Zaw, it does static analysis based only on permission request features of application. Based on permissions requested by any application, it is categorized as malware or good ware application by machine learning technique. This technique can lead to many undetected malwares. Also number of false positives is high.

IV. MATHEMATICAL MODELING AND ALGORITHM

Let S be the set {I, O, P, F, S} where,

I : Input set

O: Output set

P: Process set

F: Failure cases

S: Successful cases

1) Input set I = Set of Android applications.

2) Output set O = Application predicted as benign or malware.

3**)** Process set: P = {P1, P2, P3, P4, P5}

P1=Extraction of Application features from AndroidManifest.xml:
Extract permissions and Intent filters from Android manifest.xml. The extraction has to be done in timely manner and in constrained environment. Represent the features as sets of strings. Figure 1 depicts the extraction of Application features.

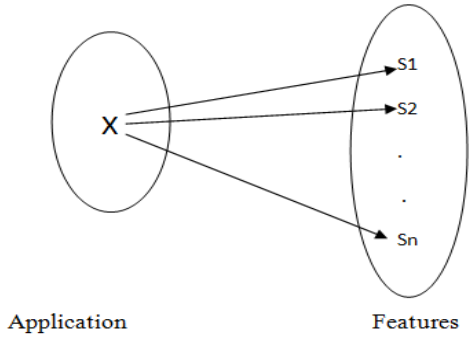

Figure 1. Mapping of application to features

P2 = Extraction of features from disassembled code.
Extract API calls from disassembled code.
P3: Assign weight to permissions, intent filters and API calls based on their malicious behavior.
P4: Fix a threshold by analysing different malware and goodware applications features with help of Virus total Service.
P5: Add weight of all the permissions, API calls and malicious intents of application to be tested. If the weight exceeds the threshold, then it is classified as malware and it is recommended to check the permissions requested by the app while installing. Check the necessity of requested permissions against the app requirement. If app doesn't need some particular access and still is requesting for it, then it is suspicious app.

4) Set F: Failure cases
The system incorrectly classifies a benign app as malware or a malware as benign app.
5) Set S: Success cases
The system is able to detect malware correctly.
NP Complete: The problem of reliably identifying a bounded-length virus is NP-complete, since the malware application can be identified and detected in fixed and short time.

V. IMPLEMENTATION STRATEGY

1. Extract permissions, Filtered intents from Android manifest.xml using PackageManager.PackageInfo. requestedPermissions.and PackageManager.queryBroadcastReceivers.
2. Extract API calls from disassembled code using dex2jar [3].

For this, extracts the classes.dex file from apk. Call dex2jar code to extract the code and API calls. The different text files with method calls are extracted. Read the methods from these text files to search the malicious API calls.

3. Assign weight to different permissions, API calls, and Filtered Intents based on their malicious behavior. Higher weights are assigned to most dangerous permissions, API calls which are frequently used in malwares [9, 2].
4. Fix a threshold of total weight of different features: permissions, API calls and filtered intents after analyzing good ware and malware applications.
5. Add the weights of different features extracted from the application and compare it with the threshold. If the weight is greater than the threshold, then application is categorized as malware else it is benign.

We classify the application with two ways:

a. Specific malware detection: Some common malwares of Android like Droid Kung Fu, Cajino baidu etc and their variants are identified.
b. Generic malware detection: Malware other than defined common malwares is identified. Any application can be scanned and analyzed to determine if it is a benign or malware.

The requested permissions and API calls are displayed which the application is using. The total weight of permissions, API calls and Intents is displayed.
Testing:

a. Download android apk from apk-downloader website [10].
b. Pass it to Virus total online application analysis tool which uses 57 antimalware tools to rate an application on a scale as good ware and malware. It gives the detection ratio.
c. Now we know if the application is benign or malware.
d. Install the apk on emulator using adb command on command prompt: adb –s emulator – 5556 install apk name with full path.
e. Once it is installed, it comes in the dropdown list of our application and can then be tested.
f. Malware samples can directly be downloaded from contagiominidump website [11].

Table 1 and table 2 lists some of the malware and benign applications tested:

DroidKungFu variants: a.com.atools.cuttherope-LeNa.b.apk b.com.rovio.new.ads-LeNa.c.apk
Cajino baidu variants: a.Cajino_5F385407A0E547F809AC4BE8B1119B04.apk b.Cajino_39581735EE24D54F93C8C51D8C39B506.apk c.Cajino_9342B4ECBB7EB045EDCDB6E0E339E415.apk d.Cajino_B3814CA9E42681B32DAFE4A52E5BDA7A.apk
_com.aijiaoyou.android.sipphone_1005_1.0.5.apk
_com.electricsheep.master.paintpro_10_2.0.1.apk
_com.sansec_9_V1.0.09.apk
_com.keji.unclear_1_1.0_BC6C20C79AED279B409C614A92 - - E63BB9.apk
Anserverb.apk
anserverb_qqgame.apk
QQ_tencent.qqgame.lord_24_1.1.apk
BloodvsZombie_com.gamelio.DrawSlasher_1_1.0.1.apk
v1.0_com.GoldDream.pg_1_1.0_F66EE5B- -8625192D0C17C0736D208B0BD.apk
AndrPJApps-Gen_ f051eeab57e42d569d298ad076c9fb47610e201e.apk

Table 1. Subset of Malware samples tested

com.shoozhoo.imageresizer.apk
me.piebridge.bible.apk
com.hansip87.smallapp.viewer.apk
com.mobikwik_new.apk
com.northpark.pushups.apk
com.onexsoftech.crackscreen.apk
com.piviandco.fatbooth.apk
com.breakingnews.apk
me.piebridge.bible.apk
com.shoozhoo.imageresizer.apk

Table 2. Subset of Benign applications tested

VI. RESULTS

Following are the parameters of evaluation:
True Positive Ratio (TPR):
TPR =TP/TP + FN
TP is the number of malware cases correctly classified.
FN is the number of malware cases misclassified as legitimate software.
False Positive Ratio (FPR):
FPR = FP/FP + TN
FP is the number of benign apps incorrectly detected as malware.
TN is the number of legitimate apps correctly classified.
Accuracy:

The total number of the classifier's hits divided by the number of instances in the whole dataset:
Accuracy = (TP + TN) *(TP + TN) / (TP + FP + TP + TN)

Comparing our system result with Permission based malware detection applying J48 classifier [2], following are the numerical results:

Detection method	TPR	FPR	Accuracy
Permission based detection.	0.87	0.25	81.32%
Detection Based on App features (Permissions + API calls)	0.88	0.20	85.0%

The TPR and Accuracy of our system are higher than that of permission based malware detection. The FPR is lower than Permission based malware detection.

VII. CONCLUSION

Our proposed method of detection of malware on Android based on application features which make use of Permissions, API calls, and Intent filters is proven to be better than only permission based malware detection. It yields better TPR and accuracy than Permission based malware detection. Most of the anti malwares in industry are signature based which maintains large database of signatures of different malwares and keep on updating the signature regularly. In comparison to this, our method is behavior based which overcomes these disadvantages and detects an application as malware or good ware based on its features. It also gives an indication to check the permissions requested by application while installation that whether they are really required or not.

VIII. ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Prof. A.K. Gupta for her exemplary guidance, monitoring and constant encouragement which helped me in completing this task through various stages. The Blessings, help and guidance given by her time to time shall carry me a long way in the journey of life which I am about to embark.

REFERENCES

[1] Zarni Aung, Win Zaw, "Permission-Based Android Malware Detection," in IJSTR Vol 2, Mar 2013
[2] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, and Gonzalo _Alvarez "PUMA: Permission Usage to detect Malware in Android" in Int. Joint Conf. Springer 2013
[3] [Online].Available: dex2jar: https://code.google.com/p/dex2jar/.
[4] (2013, Dec. 3). Smali/Baksmali: An Assembler/Disassembler for Androids Dex Format [Online]. Available:http://code.google.com/p/smali
[5] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics-aware malware detection," in Proc. IEEE Symp. Security Privacy, May 2005, pp. 3246.
[6] Yu Feng, Saswat Anand, Isil Dillig, Alex Aiken, "Apposcopy: Semantics-Based Detection of Android Malware through Static Analysis," in ACM SIGSOFT Int. Symp, November, 2014.
[7] Vaibhav Rastogi, Yan Chen, and Xuxian Jiang, "Catch Me If You Can: Evaluating Android Anti-Malware against Transformation Attacks," In IEEE Transactions, January 2014
[8] D. Arp, M. Spreitzenbarth, M. H ubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in NDSS, Internet Society, February 2014, USA.
[9] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, David Wagner, "Android Permissions Demystified," In ACM, October, 2011
[10] http://apps.evozi.com/apk-downloader/
[11] http://contagiominidump.blogspot.in/2011/07/take-sample-leave-sample-mobile-malware.html
[12] (2013, Dec. 3). Android-Apktool: A Tool for Reengineering Android APK Files [Online]. Available:http://code.google.com/p/android-apktool.