Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 794 – 803

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Spyware Detection in Android Using Hybridization of Description Analysis, Permission Mapping and Interface Analysis

Parmjit Kaur[a], Sumit Sharma[b,]*

[a] Department of Computer Science, CGC Group of Colleges, Mohali and 160055, India
[b] CSE Department, Chandigarh University, Mohali and 160055, India

Abstract

Among all available Mobile OS, Android is an ideal target for attackers due to its huge popularity. Android provides open-source OS and also provides ability to install third party applications that poses threat of user's privacy breach. In this paper, we have a close look at permissions that are granted during installation period. We have proposed a hybrid approach for detection of malicious applications by scanning with different antivirus softwares and comparing all. This hybrid approach depends upon three parameters- Description Mapping, Interface Analysis and Source Code Analysis that defines an application's behaviour i.e. either it behaves malicious or normal.

Keywords: Description;Interface Analysis;Permissions;Source Code Analysis;Spyware

1. Introduction

In recent days, Smartphones role in human life has gained significance importance. The interest of human beings in mobile computing has shifted from laptops to phones and tablets. Smartphone Market has grown with the invention of iPhone (2007) and Android phones (2008). But, an Android phone surpasses iPhone due to its Open Source Operating system, highest market share, and unrestricted application market for third party applications. Due to the advancement in functionalities of Android OS, it has been deployed by many companies like Samsung, LG, Google,

* Corresponding author. Tel.: 9464527635.
E-mail address: sumit_sharma@mailingaddress.org.

doi:10.1016/j.procs.2015.02.148

Motorola and Dell [1,2].

Security is the main issue in Android devices. Smartphones market found the first malware in 2004[3]. Trojan was the first virus which was identified by Kaspersky Lab. In 2012, a study conducted by MIT Organization in which *A.PINTO* has given a report- "*Android Malware 400% increase*" in which he had described how Android malware is increasing at high speed since Summer 2010. Malware is software which runs behind UI (user interface) and steals user's confidential data like password, bank account details and most recently visited pages etc[4].

Attacks are being introduced in the system by means of third party application through which an attacker introduce the malicious code and upload it on the Google Play Store. Infected application are being downloaded and installed by Android Users generally by allowing all the requested permissions. When user uses the infected application, in the back-ground infected app can misuse the permissions granted to it and steal the private data of the user like precise geographic location, user's contact information, and picture clicked without user knowledge etc. Due to it, attacker takes advantage of the Android Permission Model[5]. Therefore, it is important to have better understanding of Permission Based System[6].

Spyware is one of the types of malware which monitors and collects personal information about the end user like the email address, frequently visited pages, user's contact information, credit card number, key pressed by users, precise geographic location, bank account no., and picture clicked without user knowledge[7]. The objective of this research work is to detect Spyware by using hybrid approach to enhance the security of user's private data because malwares are spreading through internet into the mobile devices and provides provision for installing application from unknown sources. In this research, firstly permissions are retrieved from AndroidManifest.XML file and then it is inputted to our hybrid approach.

- We have used hybrid approach to increase the validity of detected spyware genre applications out of all the installed applications using three parameters- Description mapping, Interface Layout, and Source Code Analysis with all permissions retrieved from XML file
- We have created a dataset from extracted features of Android applications in order to develop android Malware Detection Framework
- We have developed a few applications for testing in which malicious patterns have been introduced
- We have tested each application with different anti-viruses like McAfee, Avira, Norton etc

Section 2 describes the related work, Section 3 describes an overview of Android and its permission system, Section 4 describes the Methodology to be used, Section 5 analyzes and discusses the results, and section 6 finally concludes the paper and indicates its future work.

2. Gap Analysis in Existing Work

Based on the literature review[8-18] of Android Permission Security system in Android Applications, following gaps have been identified:

- No consolidate technique have been developed to extract the English keywords used in the description of an android application.
- API documentation has been used but it does not provide sufficient information.
- There is need to check the objects and classes related to each permission defined in AndroidManifest.XML file.
- Till date no technique uses the identification of FrameLayout but it is capable of hiding surfaceview holder inside it if another child is added after surfaceview which can be used in background process to click the photos of the user and record videos etc

3. Permission Based System

Permissions are acquired by using two-way process. Firstly, developer defines the required permissions that are the first requisite for performing the functionalities of an application. Secondly, during installation time, user must have approved all the requested permissions to use an application.

The Android official Market presents each application with two installation pages. The first installation page includes install button, screenshots, description, rating and user reviews. When user click on install button, second installation page arrives that includes the application's requested permissions[9]. Lists of Permissions are requested for each application according to their level or types which are defined below:

- Level-zero (0) permissions or named as normal permissions are permissions that create low-risk factor and automatically granted by the system because these are hidden in a folded menu on the screen and typically only affect the applications scope. Example are VIBRATE and SET WALLPAPER etc[5,6]
- Level-one (1) permissions are higher-risk permissions or dangerous-level. These permissions are shown on the screen at the time of application installation[5,6]
- Level-two (2) permissions or signature permissions are based upon the criteria of key certificate matching of two or more applications such permission are utilized by developers to share resources among one or more application where the key certificate pair matches[5,6]
- Level-three (3) permissions or signatureOrSystem permission can be allowed by the system to preinstalled applications, provided by the manufactures either directly on the mobile or via updates sent over the air[5,6]

After receiving the requested lists of an application according to their level of permission, it depends upon user either they installs all application by allowing all requested permissions or denying all of the requested permissions by cancelling the installation of an application[9]. In this paper, the main focus is on security in android applications by introducing Hybrid approach in Android Permission System.

4. Methodology

In this research work, a hybrid approach has been emphasized to increase the validity of detected Spyware. We have used the android-market-api due to its capability to interact with the Google Play Store Market. In this methodology following steps have been followed which is shown in Fig. 1.

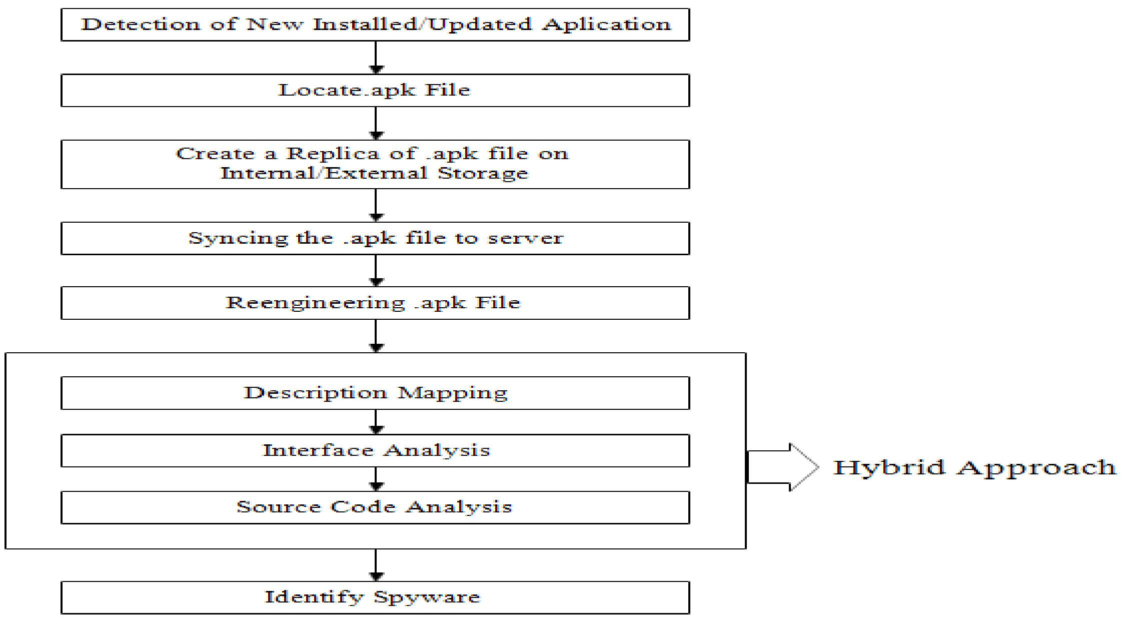

Detection of New Installed/Updated Aplication
Locate.apk File
Create a Replica of .apk file on Internal/External Storage
Syncing the .apk file to server
Reengineering .apk File
Description Mapping
Interface Analysis
Source Code Analysis
Hybrid Approach
Identify Spyware

Fig. 1. Methodology Used

There are various applications in Android Market in which some of these applications are available free of cost and some are paid ones. The main purpose of this proposed methodology is to use the Hybrid approach for permission based analysis in which three important parameters have been included that checks the malicious pattern of each application. The three parameters are Description Checking, Source Code Analysis and Interface Analysis.

To accomplish the objectives of our proposed methodology, Firstly, Broadcast Receiver is listening to the system broadcast for every newly installed application or updated application. When the receiver receives such broadcast it finds the location of .apk file which is by default hidden by the play store and then that .apk file is uploaded to the server. Re-engineering .apk file is done for each application with different tools like dex2jar, jd-gui, and apktool.

Real time server (000webhost) has been used in this approach. In this methodology, different applications are taken for analysis and that too are scanned with different antiviruses like McAfee, Avira etc. For detection, new applications are created in which malicious pattern is introduced. Permissions of the applications are retrieved from AndroidManifest.XML file and then these are compared according to the three parameters of hybrid approach as described below:

4.1 Description Mapping

When developer develops any application then description is provided about that application with the intention that user can easily understand the features and functionalities of the applications. When end-user wants to download any application from play-store, they visualize the description about that particular application. Before installing an application, users check the application on the basis of application‘s screen shots, Description, and their reviews. Analyzing the screenshot is not feasible as it involves Digital Image Processing which is computationally intensive. So, our main concentration is on analysis of description of the application because description is the first interface in user interaction that helps the end-user to understand about functionality of the application[6].

In Android applications, permissions are declared in XML File. Permissions are all or nothing based i.e. end users have privileges to install application by allowing all requested permissions or denying all of the requested permissions by cancelling the installation of that application. Information is given to end user about the permissions that application needs according to the developer of the application, but not about the permissions which the application actually uses behind the user interface[18]. Because unauthorized user can declares extra permissions for their personal use or for stealing the personal information of end-user but end-user cannot understand it. Our main purpose is to detect the Spyware according to the given description so that we can prohibit the harmful impact of unauthorized users and provide safety to end-users. This can be done by mapping the required permissions of the application with Static Permission Database. We have collected data from approximately 100 applications by checking English keywords used in their description which are related to permission defined in XML file[6].

4.2 Interface Analysis

A layout defines the visual structure for a user interface (UI) and controls how the child views are presented on the screen. There are different types of standard layouts like RelativeLayout, LinearLayout, FrameLayout, AbsoluteLayout, TableLayout[21]. Out of all these available layouts only FrameLayout is checked in this paper to detect the illegal use of camera. FrameLayout has attribute that shows only one child view at the foreground. As a result, this layout could be used to hide several others views which present the ability to gain illegitimate access of various hardware regarding the user privacy[20]. FrameLayout can be more effective when elements are unseen and shown programmatically. The attribute android:visibility is used to hide particular elements in the XML.The attribute android:visibility has three visibility values which are visible, invisible (not show, but even takes a space), and gone (not show, and not space taken) [6].

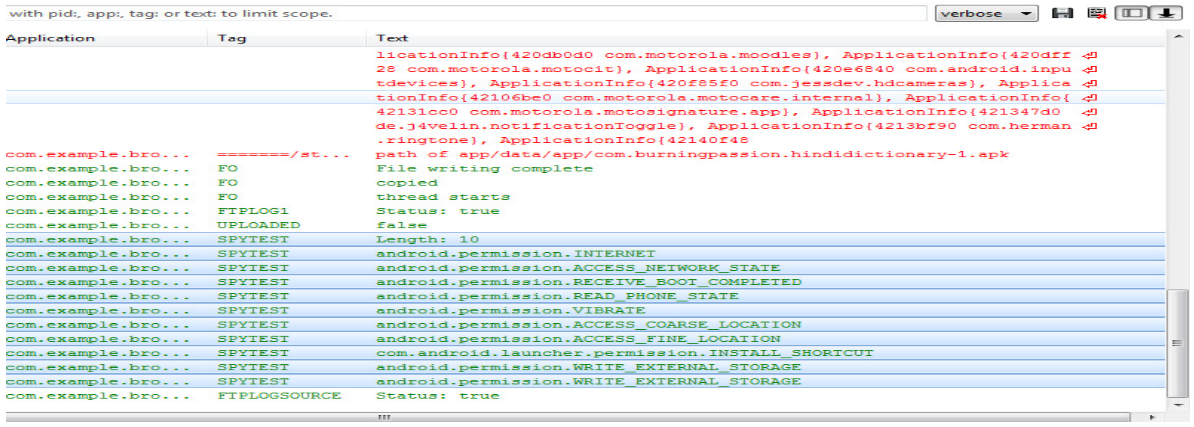
Due to its useful feature i.e. elements are unseen and displayed programmatically; unauthorized user can take the advantage of this feature and steal the personal information of user. Some malicious applications use this feature and place the Camera preview/Surface View object in a Frame Layout and place another child view to block the visibility of camera preview which may use either the front or rear camera to take user images[6].

4.3 Source Code Analysis

After reengineering .apk file, we opened the entire java files with java decompiler (jd-gui). In source code analysis, all objects and classes (declared in its java file) are checked against the permissions defined in Android.Manifest.XML. If the objects or classes related to the permissions are not found then there is a chance that malware is existed in the application, and application may steal the personal information[6].

5. Implementation

For permission based mechanism, first of all the permissions are retrieved from AndroidManifest.XML file which is shown in Fig. 2. and then these are compared according to the three parameters of hybrid approach.



```
with pid:, app:, tag: or text: to limit scope.                                   verbose

Application           Tag              Text
                                       licationInfo{420db0d0 com.motorola.moodles}, ApplicationInfo{420dff
                                       28 com.motorola.motocit}, ApplicationInfo{420e6840 com.android.inpu
                                       tdevices}, ApplicationInfo{420f85f0 com.jessdev.hdcameras}, Applica
                                       tionInfo{42106be0 com.motorola.motocare.internal}, ApplicationInfo{
                                       42131cc0 com.motorola.motosignature.app}, ApplicationInfo{421347d0
                                       de.j4velin.notificationToggle}, ApplicationInfo{4213bf90 com.herman
                                       .ringtone}, ApplicationInfo{42140f48
com.example.bro...    =======/st...    path of app/data/app/com.burningpassion.hindidictionary-1.apk
com.example.bro...    FO               File writing complete
com.example.bro...    FO               copied
com.example.bro...    FO               thread starts
com.example.bro...    FTPLOG1          Status: true
com.example.bro...    UPLOADED         false
com.example.bro...    SPYTEST          Length: 10
com.example.bro...    SPYTEST          android.permission.INTERNET
com.example.bro...    SPYTEST          android.permission.ACCESS_NETWORK_STATE
com.example.bro...    SPYTEST          android.permission.RECEIVE_BOOT_COMPLETED
com.example.bro...    SPYTEST          android.permission.READ_PHONE_STATE
com.example.bro...    SPYTEST          android.permission.VIBRATE
com.example.bro...    SPYTEST          android.permission.ACCESS_COARSE_LOCATION
com.example.bro...    SPYTEST          android.permission.ACCESS_FINE_LOCATION
com.example.bro...    SPYTEST          com.android.launcher.permission.INSTALL_SHORTCUT
com.example.bro...    SPYTEST          android.permission.WRITE_EXTERNAL_STORAGE
com.example.bro...    SPYTEST          android.permission.WRITE_EXTERNAL_STORAGE
com.example.bro...    FTPLOGSOURCE     Status: true
```

Fig. 2. Retrieval of Permission from AndroidManifest.XML file

In this approach, weightage is assigned for each parameter (20% weightage is used for description mapping, 20% weightage is used for Interface Analysis, and 60% weightage is used for source code analysis). If description is not defined for an application then Description Weightage becomes 0% i.e. the case where the user has not downloaded the application from Google Play Store and Source Weightage becomes 70% and Layout Weightage becomes 30%. Like in our case, we have created few applications in which we have introduced the malicious pattern and description is not defined in android market. At last, report is generated in SDCard for each parameter and their corresponding weightage. After calculating all the weightages, Total Weightage is calculated which is sum of all weightages. Total Weightage is compared with threshold value which is 30%. If Total Weightage is less than threshold value then application is declared as Normal application. If Total Weightage is greater than threshold value then application is declared as malicious application. Total Weightage (Total$_{\text{Weightage}}$) is calculated as

$$\boldsymbol{Total_{Weightage}} = \left(Description_{Weightage} + Layout_{Weightage} + Source_{Weightage}\right) \qquad (1)$$

For analysis, different applications are taken from android market and some new applications are also developed in which malicious code is introduced. In this paper, testing of different applications is done by considering various factors like

- Application which behaves normal
- Application which show malicious samples by testing with Avira and McAfee
- Application in which malicious code is embedded
- Application which is malicious but not detected by antivirus scanner

It has been observed that all malicious applications are declared as malicious by our hybrid approach. The main advantage of our approach is that it has shown better result than antivirus scanner and alert is sent to the user about that particular application while its installation time. Testing of each parameter and its corresponding results are shown below:

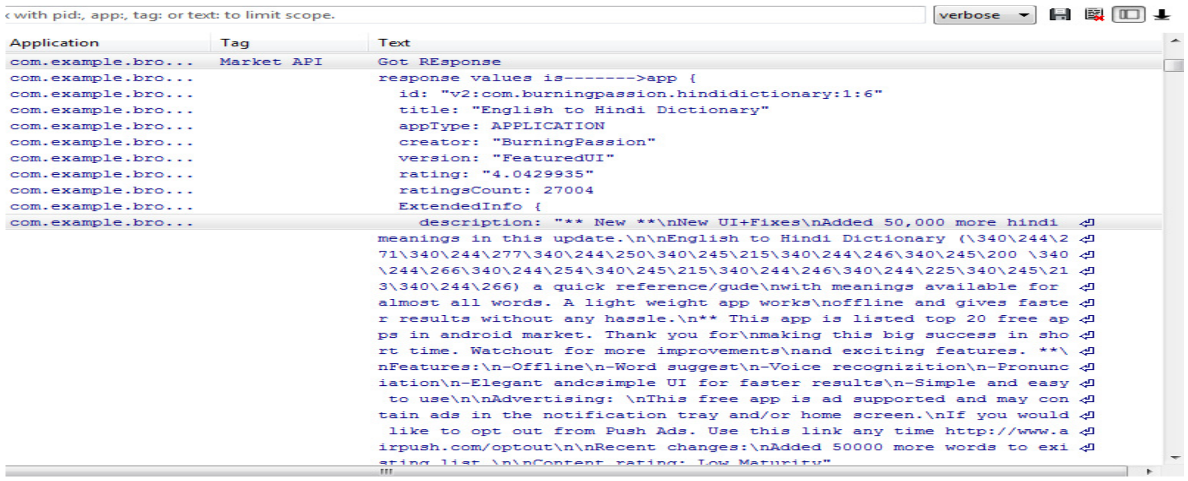
5.1 Description Mapping

Clear and well-written Description defined for each application promotes end user to identify which permission is used in that application. For analysis, database is collected from 100 android applications and same naming i.e.

english keywords is used in database as the official Android market. For that an unofficial Android Market API is used to connect with the Android market and, therefore, obtain the actual description of the applications and then compare with the permissions used in .XML file. Fig. 3. shows below the description fetched from android market. In this approach, Description Weightage ($\text{Description}_{\text{Weightage}}$) is calculated by using following formula

$$\boldsymbol{Description_{Weightage}} = \left(100 - \left(\left(\frac{\text{Total\_Description \_Count}}{\text{Total\_Permission \_Count}}\right) \times 100\right)\right) \times 0.2 \qquad (2)$$

Where Total_Description_Count describes how many keywords are matched with the total permissions used in AndroiManifest.XML file and Total_Permission_Count describes how many permissions are used in AndroidManifest.XML file.

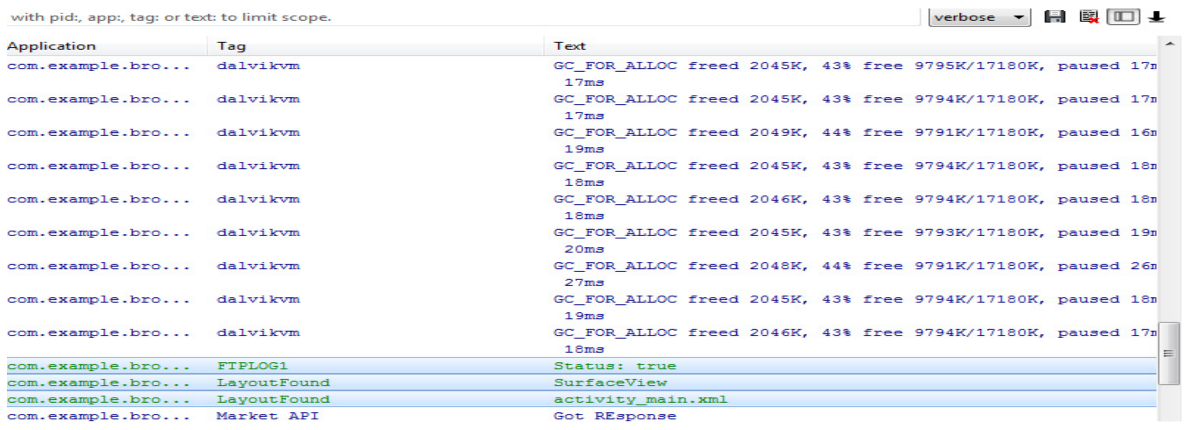


```
‹ with pid:, app:, tag: or text: to limit scope.                          verbose

Application            Tag            Text
com.example.bro...     Market API     Got REsponse
com.example.bro...                    response values is------->app {
com.example.bro...                      id: "v2:com.burningpassion.hindidictionary:1:6"
com.example.bro...                      title: "English to Hindi Dictionary"
com.example.bro...                      appType: APPLICATION
com.example.bro...                      creator: "BurningPassion"
com.example.bro...                      version: "FeaturedUI"
com.example.bro...                      rating: "4.0429935"
com.example.bro...                      ratingsCount: 27004
com.example.bro...                      ExtendedInfo {
com.example.bro...                        description: "** New **\nNew UI+Fixes\nAdded 50,000 more hindi
                                      meanings in this update.\n\nEnglish to Hindi Dictionary (\340\244\2
                                      71\340\244\277\340\244\250\340\245\215\340\244\246\340\245\200 \340
                                      \244\266\340\244\254\340\245\215\340\244\246\340\244\225\340\245\21
                                      3\340\244\266) a quick reference/gude\nwith meanings available for
                                      almost all words. A light weight app works\noffline and gives faste
                                      r results without any hassle.\n** This app is listed top 20 free ap
                                      ps in android market. Thank you for\nmaking this big success in sho
                                      rt time. Watchout for more improvements\nand exciting features. **\
                                      nFeatures:\n-Offline\n-Word suggest\n-Voice recognizition\n-Pronunc
                                      iation\n-Elegant andcsimple UI for faster results\n-Simple and easy
                                       to use\n\nAdvertising: \nThis free app is ad supported and may con
                                      tain ads in the notification tray and/or home screen.\nIf you would
                                       like to opt out from Push Ads. Use this link any time http://www.a
                                      irpush.com/optout\n\nRecent changes:\nAdded 50000 more words to exi
                                      sting list.\n\nContent rating: Low Maturity"
```

Fig. 3. Retrieval of Description from Android Market using Android Market API of English to Hindi Dictionary Application

5.2 *Interface Analysis*



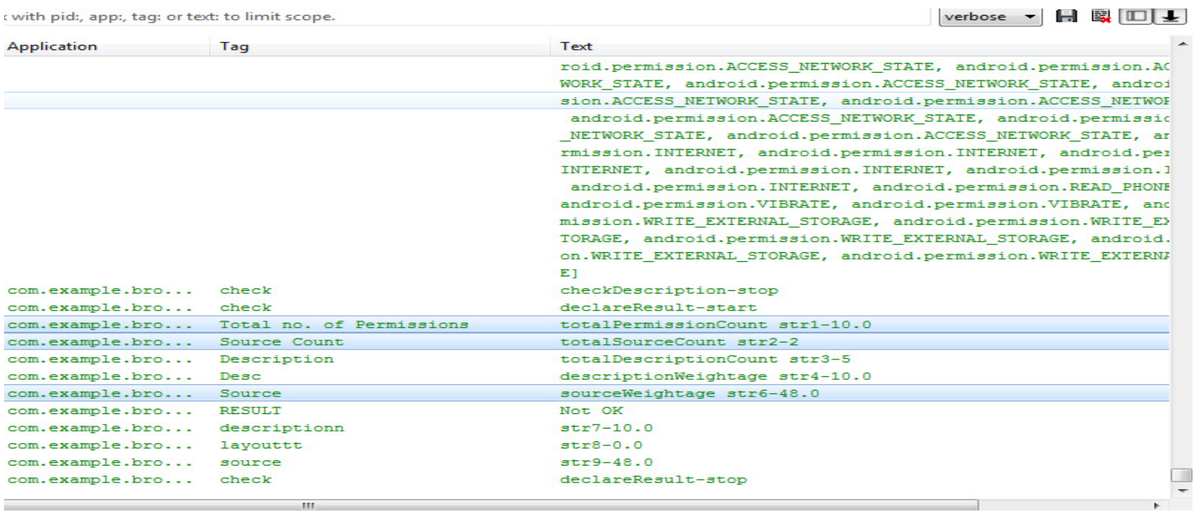
```
with pid:, app:, tag: or text: to limit scope.                            verbose

Application            Tag              Text
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2045K, 43% free 9795K/17180K, paused 17m
                                         17ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2045K, 43% free 9794K/17180K, paused 17m
                                         17ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2049K, 44% free 9791K/17180K, paused 16m
                                         19ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2045K, 43% free 9794K/17180K, paused 18m
                                         18ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2046K, 43% free 9794K/17180K, paused 18m
                                         18ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2045K, 43% free 9793K/17180K, paused 19m
                                         20ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2048K, 44% free 9791K/17180K, paused 26m
                                         27ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2045K, 43% free 9794K/17180K, paused 18m
                                         19ms
com.example.bro...     dalvikvm         GC_FOR_ALLOC freed 2046K, 43% free 9794K/17180K, paused 17m
                                         18ms
com.example.bro...     FTPLOG1          Status: true
com.example.bro...     LayoutFound      SurfaceView
com.example.bro...     LayoutFound      activity_main.xml
com.example.bro...     Market API       Got REsponse
```

Fig. 4. Activity_Main.XML Layout found that uses Malicious Code

In this analysis, all layout files which are located in res folder are checked for the services which they perform at the background without knowledge of user. In this parameter, android:visibility attribute is checked in all layout files in which FrameLayout is used. If value of android:visibility is invisible or gone than it is treated as Malicious app. Layout Weightage ($\text{Layout}_{\text{Weightage}}$) becomes 20% if it found frame layout and their corresponding visibility invisible or gone otherwise zero. Fig. 4. shows above the layout found and relative their name of that layout in which malicious code is found.

5.3 Source Code Analysis



```
: with pid:, app:, tag: or text: to limit scope.                          verbose

Application           Tag                         Text
                                                  roid.permission.ACCESS_NETWORK_STATE, android.permission.AC
                                                  WORK_STATE, android.permission.ACCESS_NETWORK_STATE, androi
                                                  sion.ACCESS_NETWORK_STATE, android.permission.ACCESS_NETWOF
                                                   android.permission.ACCESS_NETWORK_STATE, android.permissic
                                                  _NETWORK_STATE, android.permission.ACCESS_NETWORK_STATE, ar
                                                  rmission.INTERNET, android.permission.INTERNET, android.pe
                                                  INTERNET, android.permission.INTERNET, android.permission.I
                                                   android.permission.INTERNET, android.permission.READ_PHONE
                                                  android.permission.VIBRATE, android.permission.VIBRATE, anc
                                                  mission.WRITE_EXTERNAL_STORAGE, android.permission.WRITE_EX
                                                  TORAGE, android.permission.WRITE_EXTERNAL_STORAGE, android.
                                                  on.WRITE_EXTERNAL_STORAGE, android.permission.WRITE_EXTERNA
                                                  E]
com.example.bro...    check                       checkDescription-stop
com.example.bro...    check                       declareResult-start
com.example.bro...    Total no. of Permissions    totalPermissionCount str1-10.0
com.example.bro...    Source Count                totalSourceCount str2-2
com.example.bro...    Description                 totalDescriptionCount str3-5
com.example.bro...    Desc                        descriptionWeightage str4-10.0
com.example.bro...    Source                      sourceWeightage str6-48.0
com.example.bro...    RESULT                      Not OK
com.example.bro...    descriptionn                str7-10.0
com.example.bro...    layouttt                    str8-0.0
com.example.bro...    source                      str9-48.0
com.example.bro...    check                       declareResult-stop
```

Fig. 5. Representation of Total_Source_Count and Source Weightage

In this analysis, objects and classes are mapped with the permission defined in the AndroidManifest.XML file. Fig. 5. shows above the Total_Source_Count and Total_Permission_Count and their corresponding Source Weightage. Source Weightage ($Source_{weightage}$) is calculated using following formula

$$\boldsymbol{Source_{Weightage}} = \left(100 - \left(\left(\frac{Total\_Source\_Count}{Total\_Permission\_Count}\right) \times 100\right)\right) \times 0.6 \tag{3}$$

Where Total_Source_Count describes how many objects and classes are found with respect to the total permission used in AndroiManifest.XML file and Total_Permission_Count describes how many permission is used in AndroidManifest.XML file. Table 1 shows the comparison of our hybrid approach to the different antivirus scanner like Avira, McAfee, Norton, AVG Antivirus etc and our methodology shows the better results.

Table 1 Comparison of Our Hybrid Approach to the different Antivirus Scanner

Apps Name	*Avira*	*McAfee*	*Norton*	*AVG*	*Avast*	*Quick Heal*	*Lookout*	*Perm. Watcher*	*Proposal Approach*
English to Hindi Dict.	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*No*	*Yes*	*Yes*
Magic Photo Effect	*Yes*	*Yes*	*No*	*Yes*	*No*	*Yes*	*No*	*No*	*Yes*
Indian Caller Info	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*No*	*Yes*	*Yes*
View Finder	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*Yes*	*Yes*
Doge Coin	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*No*
Testing 1	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*Yes*
Testing 2	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*No*	*Yes*

Here, No represents that the applications behave normally or does not contain malicious pattern and Yes represents the applications that contain malicious patterns.

6. Conclusion and Future Work

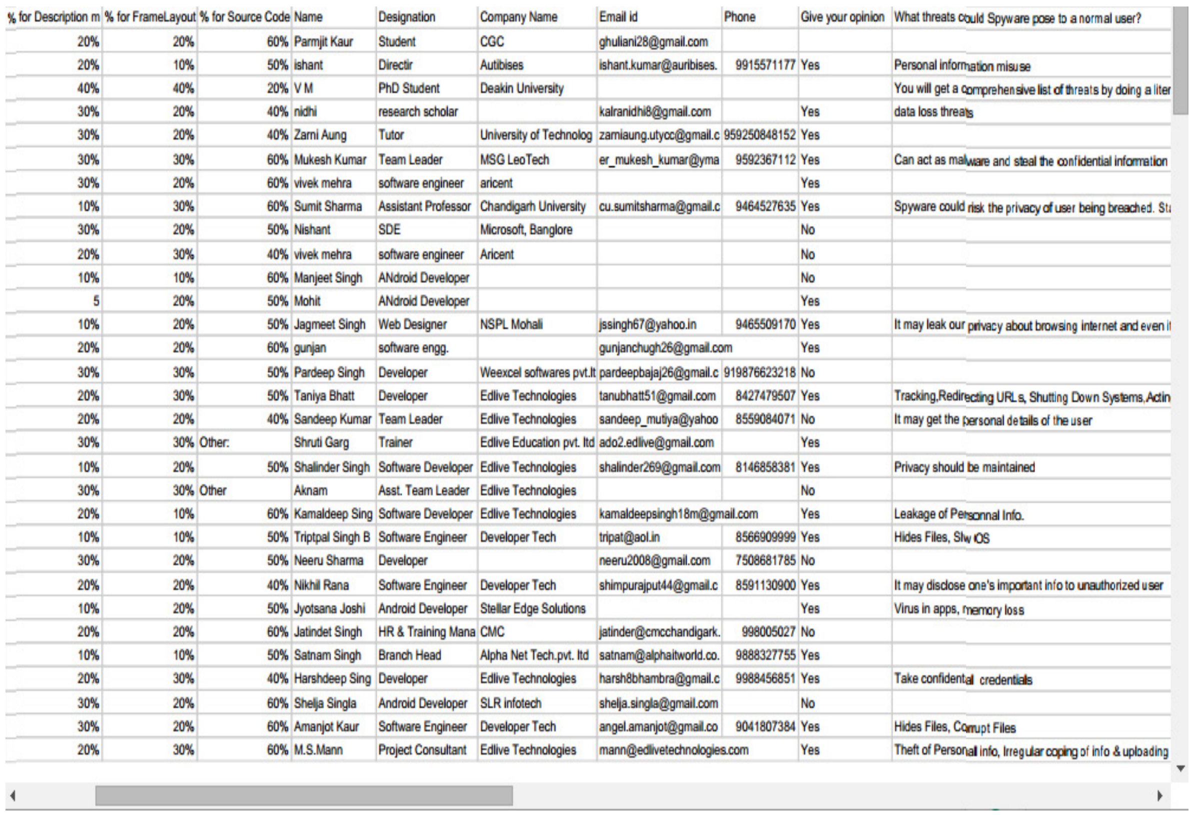
In this paper, we have defined the need of why we are doing this analysis. We have used the hybrid approach to increase the validity of detected Spyware in which analysis is done on the basis of three parameters named Description Mapping, Interface Analysis, and Source Code Analysis. We have concluded that our hybrid approach has given better results than antivirus scanner because the other system can't find the malicious applications in which Interface Analysis is done and on the basis of description of the applications.
The future work in this domain involves selective granting of permissions at the installation time. Along the description mapping, the user rating provided for the app could be used to evaluate the possibility of spyware existence in the app as well as the user reviews provided could be analyzed with the help of NLP.

Acknowledgements

The help rendered by Alpha IT, Amdocs, Aricent, Auribises, Developer Tech, Edlive Technologies, Microsoft, MSG Leo Tech, NSPL Mohali, Phd student of Deakin University, Samsung, Stellar Edge, SLR Infotech, and University of Technology for experimentation is greatly acknowledged.

Appendix A. Weightage Assignment for Hybrid Approach

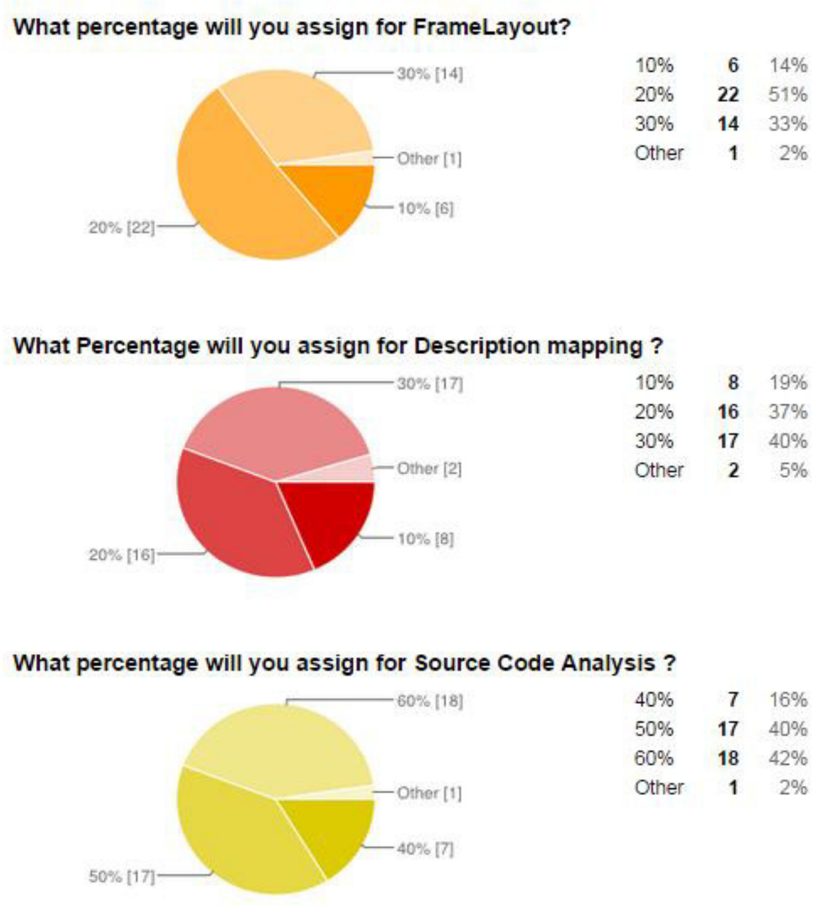
The data is collected from different developers for assigning weightage to three parameters used in Hybrid approach in which multiple questions are answered by developers and their corresponding answers are shown below:



% for Description m	% for FrameLayout	% for Source Code	Name	Designation	Company Name	Email id	Phone	Give your opinion	What threats could Spyware pose to a normal user?
20%	20%	60%	Parmjit Kaur	Student	CGC	ghuliani28@gmail.com			
20%	10%	50%	ishant	Directir	Autibises	ishant.kumar@auribises.	9915571177	Yes	Personal information misuse
40%	40%	20%	V M	PhD Student	Deakin University				You will get a comprehensive list of threats by doing a liter
30%	20%	40%	nidhi	research scholar		kalranidhi8@gmail.com		Yes	data loss threats
30%	20%	40%	Zarni Aung	Tutor	University of Technolog	zarniaung.utycc@gmail.c	959250848152	Yes	
30%	30%	60%	Mukesh Kumar	Team Leader	MSG LeoTech	er_mukesh_kumar@yma	9592367112	Yes	Can act as malware and steal the confidential information
30%	20%	60%	vivek mehra	software engineer	aricent			Yes	
10%	30%	60%	Sumit Sharma	Assistant Professor	Chandigarh University	cu.sumitsharma@gmail.c	9464527635	Yes	Spyware could risk the privacy of user being breached. St
30%	20%	50%	Nishant	SDE	Microsoft, Banglore			No	
20%	30%	40%	vivek mehra	software engineer	Aricent			No	
10%	10%	60%	Manjeet Singh	ANdroid Developer				No	
5	20%	50%	Mohit	ANdroid Developer				Yes	
10%	20%	50%	Jagmeet Singh	Web Designer	NSPL Mohali	jssingh67@yahoo.in	9465509170	Yes	It may leak our privacy about browsing internet and even i
20%	20%	60%	gunjan	software engg.		gunjanchugh26@gmail.com		Yes	
30%	30%	50%	Pardeep Singh	Developer	Weexcel softwares pvt.lt	pardeepbajaj26@gmail.c	919876623218	No	
20%	30%	50%	Taniya Bhatt	Developer	Edlive Technologies	tanubhatt51@gmail.com	8427479507	Yes	Tracking,Redirecting URLs, Shutting Down Systems,Actin
20%	20%	40%	Sandeep Kumar	Team Leader	Edlive Technologies	sandeep_mutiya@yahoo	8559084071	No	It may get the personal details of the user
30%	30%	Other:	Shruti Garg	Trainer	Edlive Education pvt. ltd	ado2.edlive@gmail.com		Yes	
10%	20%	50%	Shalinder Singh	Software Developer	Edlive Technologies	shalinder269@gmail.com	8146858381	Yes	Privacy should be maintained
30%	30%	Other	Aknam	Asst. Team Leader	Edlive Technologies			No	
20%	10%	60%	Kamaldeep Sing	Software Developer	Edlive Technologies	kamaldeepsingh18m@gmail.com		Yes	Leakage of Personnal Info.
10%	10%	50%	Triptpal Singh B	Software Engineer	Developer Tech	tripat@aol.in	8566909999	Yes	Hides Files, Slw iOS
30%	20%	50%	Neeru Sharma	Developer		neeru2008@gmail.com	7508681785	No	
20%	20%	40%	Nikhil Rana	Software Engineer	Developer Tech	shimpurajput44@gmail.c	8591130900	Yes	It may disclose one's important info to unauthorized user
10%	20%	50%	Jyotsana Joshi	Android Developer	Stellar Edge Solutions			Yes	Virus in apps, memory loss
20%	20%	60%	Jatindet Singh	HR & Training Mana	CMC	jatinder@cmcchandigark.	998005027	No	
10%	10%	50%	Satnam Singh	Branch Head	Alpha Net Tech.pvt. ltd	satnam@alphaitworld.co.	9888327755	Yes	
20%	30%	40%	Harshdeep Sing	Developer	Edlive Technologies	harsh8bhambra@gmail.c	9988456851	Yes	Take confidental credentials
30%	20%	60%	Shelja Singla	Android Developer	SLR infotech	shelja.singla@gmail.com		No	
30%	20%	60%	Amanjot Kaur	Software Engineer	Developer Tech	angel.amanjot@gmail.co	9041807384	Yes	Hides Files, Corrupt Files
20%	30%	60%	M.S.Mann	Project Consultant	Edlive Technologies	mann@edlivetechnologies.com		Yes	Theft of Personal info, Irregular coping of info & uploading

Fig. A-1. Data collected from different developers during Survey

In the next figure, graph is drawn on the basis of our hybrid approach in which three parameters are used: Description Mapping, Interface Layout, Source Code Analysis and their corresponding data is collected from different developers which is previously shown in excel sheet.



What percentage will you assign for FrameLayout?

10%	6	14%
20%	22	51%
30%	14	33%
Other	1	2%

What Percentage will you assign for Description mapping ?

10%	8	19%
20%	16	37%
30%	17	40%
Other	2	5%

What percentage will you assign for Source Code Analysis ?

40%	7	16%
50%	17	40%
60%	18	42%
Other	1	2%

Fig. A-2. Graphical Representation of Data collected related to Three Parameters

References

1. Kaur P, Sharma S. Google Android a mobile platform: A review, In *Recent Advances in Engineering and Computational Sciences*; 2014, IEEE, p. 1-5.
2. Kirandeep, Garg A. Implementing Security on Android Application, In *The International Journal of Engineering and Science;* 2013, 2, p. 56-59.
3. Schmidt AD, Schmidt HG, Batyuk L, Clausen JH, Camtepe SA, Albayrak S, Yildizli C. Smartphone malware evolution revisited: Android next target?, In *2009 4th International Conference on Malicious and Unwanted Software*; IEEE, p. 1-7.
4. Pinto A. 2012 *Android Malware 400% increase*, Available from: <http://cybersecurity.mit.edu/2012/11/android-malware-400-increase/>. 4 November 2012.
5. Satpal, Nitin B. *Enhancing Permission Model of Android*. PhD Thesis, Indian Institute of Technology Bombay; 2013.
6. Sharma S, Kaur P. Spyware Detection based upon Hybrid Approach in Android Smartphones, In *International Journal of Computer Applications,* 98, July 2014.
7. Vinod R, Jaipur R, Laxmi V, Gaur M. Survey on malware detection methods, In *Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security*; 2009, p. 74-79.

8. Eric S, Seifert J, Ullenbeck S, Rukzio E, Wolf C. Permission Watcher: Creating User Awareness of Application Permissions in Mobile Systems, In *proceedings 2012 Springer*, p. 65-80.
9. Vidas T, Christin N, Cranor L. Curbing android permission creep, In *Proceedings of the Web,* 2011, 2.
10. Blasing T, Batyuk L, Schmidt A, Camtepe SA, Albayrak S. An android application sandbox system for suspicious software detection, In *5th International Conference on Malicious and Unwanted Software;* 2010, IEEE, p. 55-62.
11. Nauman M, Khan S, Zhang X. Apex: extending android permission model and enforcement with user-defined runtime constraints, In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, 2010, p. 328-332.
12. Kern M, Sametinger J. Permission Tracking in Android, In *The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies UBICOMM;* 2012, p. 148-155.
13. Yang L, Boushehrinejadmoradi N, Roy P, Ganapathy V, Iftode L. Short paper: enhancing users' comprehension of android permissions, In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, 2012, p. 21-26.
14. Johnson R, Wang Z, Gagnon C, Stavrou A. Analysis of Android Applications' Permissions, in *IEEE Sixth International Conference on Software Security and Reliability Companion;* 2012, p. 45-46.
15. Aung Z, Zaw W. Permission-based Android malware detection, In *International Journal of Scientific & Technology Research* 2, 2013.
16. Sanz B, Santos I, Laorden C, Pedrero X, Bringas PG, Álvarez G. Puma: Permission usage to detect malware in android, In *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*, 2013, p. 289-298. Springer Berlin Heidelberg.
17. Felt AP, Elizabeth H, Egelman S, Haney A, Chin E, Wagner D. Android permissions: User attention, comprehension, and behavior, In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, 2012, ACM, p. 3.
18. Lee S, Young DJ. Assessment of malicious applications using permissions and enhanced user interfaces on Android, In *IEEE International Conference on Intelligence and Security Informatics; 2013*, p. 270-279.
19. Kathy A, Yee W, Zhou YF, Huang Z, Gill P, Lie D. Short paper: a look at smartphone permission models, In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, p. 63-68.
20. *Learn Android- Tutorials For Developing With Android, 2010.* Available from: <http://www.learn-android.com/2010/01/05/android-layout-tutorial/>. 5 January 2010
21. *The Android Developer's Guide – Frame Layout*, 2014. Available from: <http://developer.android.com/reference/android/widget/FrameLayout.html>. 23 July 2014