International Journal of Engineering Research and General Science Volume 4, Issue 3, May-June, 2016
ISSN 2091-2730

Exploring the Malicious Android Applications and Mitigating Risk using Static Analysis

Kavitha.K
PG student, Department of Computer Science and Engineering, Pondicherry Engineering College Puducherry-605014, India
kindkavi@pec.edu

Salini.P
Assistant Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry-605014, India
salini@pec.edu

Abstract— Android is a most prevalent OS for smart phones, which 84.4% of people use it as a part of life for both personal and commercial use. The usage of the android phones increases day by day, in which making the world connected through a worldwide. The android bring a drastic change in a today's modern world. As a part of it, Android was designed with much security to which malware, adware and other security issues cause a great problem in the system. Hence we propose a system to protect against these vulnerable issues and to protect our private data and to use the trusted system. Hence, it involving the two main phases consisting of detection phase and prevention phase. The detection phase involving the identifying the vulnerable issues such as identifying malware and risk based on different permission. The prevention phase involving the removal of security issues and reduced usage of permission.

Keywords—Malicious application, Android, detection, prevention, Android security, vulnerabilities, static analysis and permission

INTRODUCTION

Android operating system (OS) is mainly familiarized with lot of encroachment and its diverged features which make people towards the smarter and revolutionized world. The growth of android usage increases day by day, in which the android mobile operating system (OS) is Google's open and free software stack that finds includes middleware, and also crucial applications for the mobile devices, including smartphones [19]. The increase in the sale of android in the global market as per on 2016 statistics resulted to 83.4%when compared all other OS. It's no doubt that the apps are extremely acquainted in the market for their astonishing features and the wonderful benefits of android apps makes the users to get it. When concerned about security, malware protection is a major issue in which Android has been a major mark which a great threat leading to malicious applications (malapps).

In today's modern world android play a crucial role .The people gets moved to android for its amazing features such as portability and mobility. In android permission control is the essential point in which it restricts the access of an application. The defensive layers of security are shown in the Figure1 [20].

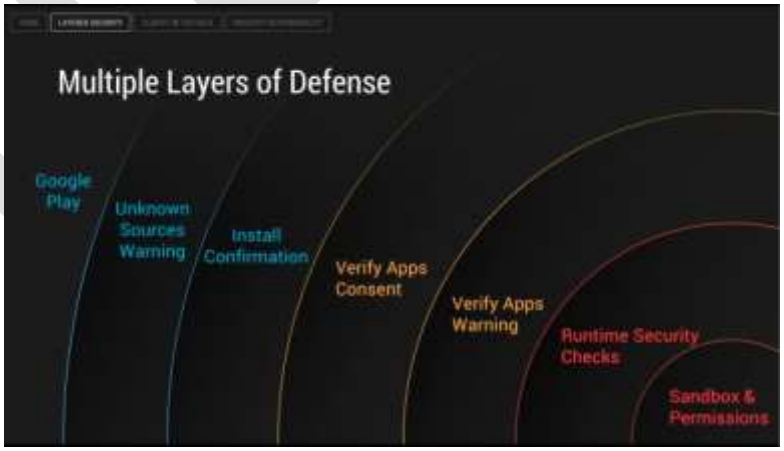


Figure1-Multiple Layer Of Defense

The main area focused here is the sandbox and permission. Generally, a sandbox is a security mechanism which is isolated from running programs. It is often used for testing the system for third parties to use authentication

Therefore, when an app needs to install the user has to check the permission and need to mount. Thus the people attracted to the amazing features of an app and app users unaware about an EULA [End Users License Agreement] .In order, to provide security and to make private information from leaking the system is designed with these robust security architectures and rigorous security programs.

II. RELATED WORKS

In this section, it intends a survey about the android security, how the malicious application is detected and its various classifications. The main areas focus on the analysis of malware application and risk detection rates.

On a survey analysis, the android users increase day by day and till date the development of android is faster which reached the N series with attractive features.

The survey [17] reported that increase in usage of the application, the attackers cause great threat to the users, which results in the number of vulnerable activities resulting the adware ,malware, spyware, etc..,

The maliciousness of android can be identified by using the static analysis, which involves the identification and detection of malapps .It also includes the classification of risk according to the vulnerabilities present in it [1].In another work, it involves the analyzing the malicious application by examining the source code that is by dynamic analysis, which is processed by vetting the behaviors of apps [2].

In includes a methodology, adding the metadata feature of creator information [6] of which along with static analysis .It also helps in distinctive the malapps and benign apps easily.

The automatic taint propagation [5] involving the automated testing of android which produces better effectiveness and performance .It is a more effective performance in detection of testing, which results with behavior analysis with the emulator.

A technique involving the permlyzer [7], which comprises the combination of runtime analysis and static to obtain automatically analyzes of permission. It provides the characteristic of each application involving all types of application. With a deep investigation of a various work, an idea to overcome the drawbacks in the proposed work.

III. PROPOSED FRAMEWORK

Our proposed work is introduced in order to minimize the risk permission which preventing the private data from leakage. It also includes the detection rate of risk and reduction of permissions in which limiting the leakage of information. Hence, in considering upon an installation of the app, permissions play a key role in it. Many users unaware of installation permission where users ignorant of the malicious threats. Lots of unnecessary permission leaks to the unauthorized users where leading a malicious activity. So, it involves a framework of detection of malicious application and prevention of data (minimization of risk).The proposed architecture is as follows:

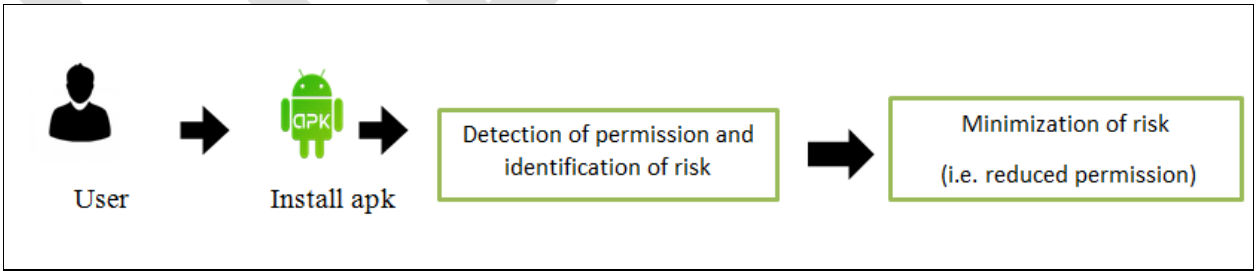


Figure 2- Architectural Design of proposed work

Here we describe the stepwise description of each module. The modules involved:

1. Installation and Initialization phase
2. Detection of risk
3. Minimization of risk
4. Result analysis with reduced risk.

1) Installation and Initialization phase

This is the first step of the proposed work in which user has to install all the necessary files where the app can be processed and initiated. It process involved is explained below.

User: User is the first access point in which enters the login IDs during the fixing particular app. The User has to check the account details of every app, and read the footings and settings of an app before entering the credentials. To protect that value the platform proposing an application atmosphere that should safeguard the security of users, data, applications, the device, and the network.

Installation of an app:

Next stage involving the installation of an app in which for an downloaded application (.apk file) the user has to verify the EULA of a particular app to be downloaded. Every app requires an permission. The permission inquiring the private info's leaks the data protruding to the vulnerable attacks. After verification, the app can be installed.

2) Detection of risk

In this module it detects the permissions of an installed app. If the user installed app, the user accepts the permission it which approved to the terms and conditions of an app. Once installed app granted permission cannot deny. So it enumerates the permissions of an installed app[19].

Detection permission Identification

Once the permission is listed it recognizes the private risk permission and examines it .Here the credentials of the permission are grouped according to type of permission. It categorizes the type of permission and list the malicious app detection.

3) Minimization of risk

In order to lessen the risk permission (i.e.) preclusion of private leakage we use the centralized algorithm which is used for better effectiveness performance. The algorithm used for reducing the risk is defined as follows:

Centralized algorithm

- Coordinator(System)

```
loop
Receive (msg);
case msg of
REQUEST: if anybody in cs
        then reply GRANTED
        else queue the REQ;
        reply DENIED.
RELEASE: if queue not empty then
          Remove the 1st on queue
          Reply GRANTED
end case
end loop
```

- Client

```
send (REQUEST);
receive (msg);
if msg! =GRANTED then receive msg;
enter cs;
send (RELEASE).
```

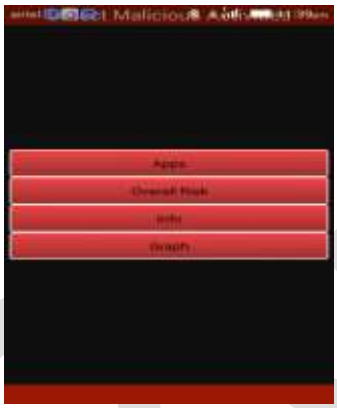
4) Result analysis with reduced risk

From the above systems the users isolate unwanted permission and shorten the risk permission which to attain a trusted permission avoiding the leakage of data. In this it yields a lesser amount of permission of an each app inhibiting the leakage of data evading over privileged permission.

From our analysis, a substantial number of apps request access the permissions in which the users get the maximum number of trusted apps which free from vulnerable threats and issues.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

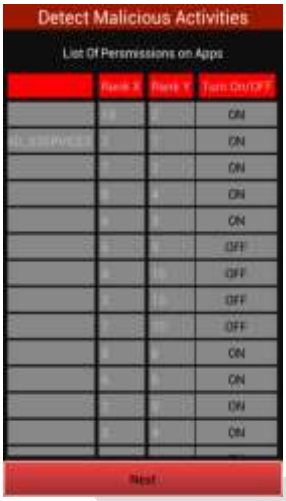
In this, it displays the snapshots on the mobile view showing an overall menu of the app.in which displays set of all menus. It identifies the all list of installed apps where if the application gets selected it identifies the list of permissions.



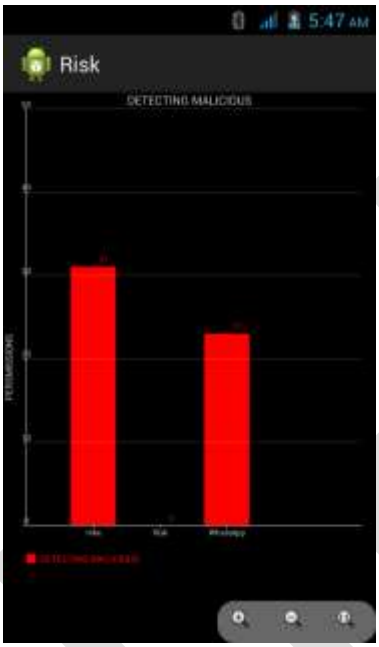
Then, it shows the menu of a risk app displaying the menu of an app. It contains the menu button of apps, overall risk, info and graphical view .In the menu, select apps button it shows the list of installed apps. This is displayed in the following figure.

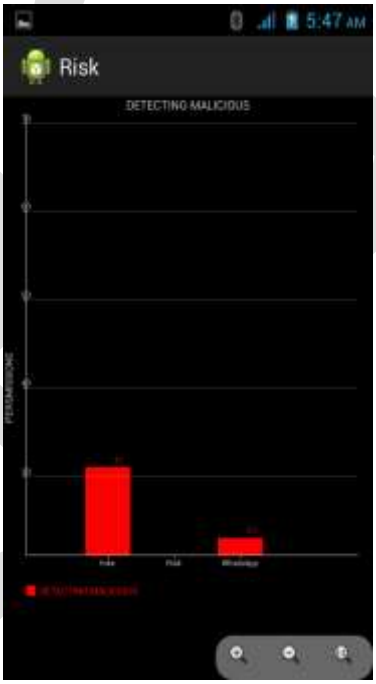
Then the selection of app is performed .For checking the permission of an app, it has been selected .The permission of an app is analyzed by tapping the icon twice. The selection of an app shown and the list off permissions to be performed is shown in the figure.

Next involves the button creation in which selection of permission based on user request which also involving the ranking of permission which includes the grouping of permissions. The centralized algorithm involves the permission control and access of the permission. The display of the controlled permission is displayed in the Figure



Then , the comparative results of the overall permission with high risk and the low risk shown in below figure.





The graph for each algorithm on each permission comparing the values with the below graph shows categorical view on algorithm where the graph of each app analyzed and retrieved risk of an each app showing the detection algorithm of each app and results.

V. EVALUATION METRICS

The performance of the proposed framework is measured in terms of the quality measures namely Accuracy, F-score and T-value[19]. Therefore the metrics are categorized as follows:

1. Accuracy
2. F-Score&
3. T-value

1. Accuracy:

Accuracy is also defined as the fraction of risk permission that is relevant. Accuracy is also defined as the proportion of correctly classified data objects both True positives (TP) and True negatives (TN) in the population (sum of TP, TN, as well as false positives (FP) and False Negatives (FN)):

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}$$

2. F-Score is defined as the fraction of lies between precision and recall where, Precision is positive predictive (fraction of retrieved permissions that are relevant).Recall is also defined as fraction of the permission that is relevant to the query that is successfully retrieved.

$$Precision = \frac{TP}{(TP + FP)}$$

And, Recall is defined as

$$Recall = \frac{TP}{TP + FN}$$

$$F - Score = 2.\frac{Precision.Recall}{Precision+Recall}$$

3. **T-value:** T-value is mainly used for better accuracy results in which highly to reduce false positives. It can be calculated using the formula,

$$t = \frac{\mu_0 - \mu_1}{\sigma_{01}\sqrt{\frac{1}{N_0} + \frac{1}{N_1}}}$$

Where, N0, μ0 and σ0 be the number, the mean and the standard deviation of X in benign samples.N1, μ1 and σ1 be the number, the mean and the standard deviation of X in malicious samples.

Finally, the results are shown the above graph and Evaluation metrics is calculated. The experiments are conducted to assess the performance of each app. The Table 5.1 showing the Accuracy value for different set of inputs.

METHODS	SAMPLE INPUTS	TP	TN	FP	FN	ACCURACY
T-TEST	1-10	4	2	2	2	.6
	11-20	5	2	1	2	.7
	21-30	4	3	1	2	.7

TABLE 5.1 Accuracy Results for T-Test

Next, the following Table 5.2 shows the assumed values for inputs calculating the shows the precision [True positive rate (TPR)] and recall [False Positive Rate [FPR]] in which calculating the F-score value for finding the performance.

SL NO	METHOD	PRECISION	RECALL	F-SCORE
1	MI	.9228	.0059	0.7
2	Corr.Coeff	.9232	.0060	0.71
3	T-Test	.9230	.0060	0.69
4	SFS	.9226	.0059	0.86
5	PCA	.9258	.0056	0.8
6	RF	.9281	.0059	0.87
7	D-tree	.9462	.0060	0.85
8	Centralized	.9520	.0062	0.9

TABLE 5.2 Comparative Detection Results

.Then includes the graphical view of F-score of different methods which is displayed below.

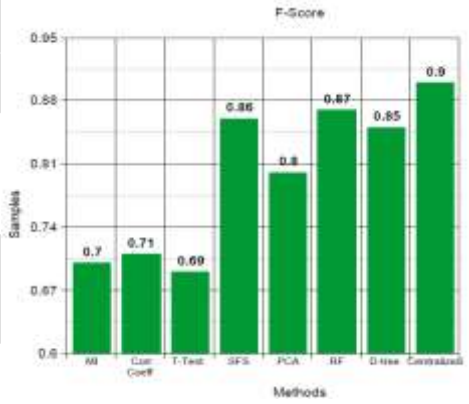


Table 5.3 F-Score Comparison

Then includes the key features of the proposed work in which analyzing the different qualities to get an overall description of various methods. This table analyzes the permission, algorithms, techniques involved and performance analysis this is displayed in the Table 5.4

Topic	Features
1.PERMISSION	Detection of permission of an app and Reduction of permission
2..SECURITY	It only identifies the risk and it provides security, which prevents the leakage of information
3.ALGORITHMS	Centralized Algorithm
4.TRUSTNESS	It indicates the risk level and its more trusted where no rooting is necessary
5.ACCURACY	Accuracy is high when compared to other

Table 5.4 Key Features with the proposed work

ACKNOWLEDGMENT

I express the sincere thanks to our Institution for extending the infrastructural facilities to carry our work successful.

CONCLUSION

This paper, involving the exploration of the malicious application and identifying the risk level which help in reducing the risk level, which help users to use the trusted app. The app makes the users protected app where the private information is not leaked.

REFERENCES:

[1] Yuan Zhang, Min Yang, Zhemin Yang, Guofei GU, Peng Ning, and Binyu Zang "Exploring Permission Induced Risk in Android–Applications for Malicious Detection" In IEEE transactions on information forensics and security, vol. 9, no. 11, November 2014.

[2] Yuan Zhang, Min Yang, Zhemin Yang, and Binyu Zang.in "Permission Use Analysis for Vetting Undesirable Behaviors in Android Apps"In IEEE transactions on information forensics and security, vol. 9, no. 11, November 2014.

[3] Sebastian Salva · Stassia R. Zafimiharisoa "APSET-an Android application Security Testing tool for detecting intent-based vulnerabilities" Published online: 27 February 2014 © Springer-Verlag Berlin Heidelberg 2014.

[4] Saurabh Chakradeo, Bradley Reaves, and Patrick Traynor "MAST: Triage for Market-Scale Mobile Malware analysis" Available Online in:http://www.cc.gatech.edu/~traynor/pap

[5] Zhu Kelong, Song Yubo, "Implementation Of Automated Testing System For Android Applications Based On Dynamic Taint Propagation" ,Chen Fei School of information science and engineering, Southeast University, Nanjing 210096, China.

[6] Hyunjae Kang,Jae-wook Jang, Aziz Mohaisen and Huy Kang Kim "Detecting and Classifying Android Malware Using Static Analysis along with Creator Information" Hindawi Publishing Corporation, International Journal of Distributed Sensor Networks Volume 2015, Article ID 479174, 9 pages.

[7] Wei Xu, Fangfang Zhang, and Sencun Zhu "Permlyzer: Analyzing Permission Usage in Android Applications" Department of Computer Science and Engineering Pennsylvania State University Park, PA.

[8] M. C. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker: Scalable and accurate zero-day Android malware detection," in Proc. 10th Int. Conf. MobiSys ,pp. 281–294, 2012.

[9] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of Android application security," in Proc. 20th USENIX Secur. Symp, p. 21-31,2011.

[10] B.Uscilowski "Mobile Adware and Malware Analysis" Available online at: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_ and_malware_analysis.pdf, Symantec White Paper(2013).
[11] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin, "Permission re-delegation: Attacks and defenses," in Proc. 20th USENIX Conf. Secur. Symp. pp. 22-28, 2011.
[12] N. Peiravian and X. Zhu, "Machine learning for Android malware detection using permission and API calls," in Proc. IEEE 25th ICTAI, pp. 300–305, Nov. 2013.
[13] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets," in Proc. NDSS Symp., pp. 1–13, 2012.
[14] M. Grace, Y. Zhou, Z. Wang, and X. Jiang, "Systematic detection of capability leaks in stock Android smartphones," in Proc. NDSS Symp., 2012, pp. 1–15.
[15] W. Enck et al., "TaintDroid: An information flow tracking system for real-time privacy monitoring on smartphones," in Proc. 9th USENIX Conf. OSDI, pp. 1–6, 2010.
[16] http://www.engineersgarage.com/articles/what-is-android-introduction.
[17] https://en.wikipedia.org/wiki/Android_(operating_system).
[18] https:www.makeuseof.com/tag/app-permissions-work-care-android/
[19] K.Kavitha,P.Salini and V.Ilamathy in the "Exploring the Malicious Android Applications and Reducing Risk using Static Analysis" in Proc. of International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)2016, pp.1316-1319.
[20] https://psadda.wordpress.com/2015/02/26/android-security-a-big-question/