ANDROID UNTRUSTED DETECTION WITH PERMISSION BASED SCORING ANALYSIS

Jackelou Sulapas Mapa

College of Information Technology,Saint Joseph Institute of Technology,Montilla Boulevard, Butuan City.

ABSTRACT

Android smart phone is one of the fast growing mobile phones and because of these it the one of the most preferred target of malware developer. Malware apps can penetrate the device and gain privileges in which it can perform malicious activities such reading user contact, misusing of private information such as sending SMS and can harm user by exploiting the users private data which is stored in the device. The study is about implementation of detecting untrusted on android applications, which would be the basis of all future development regarding malware detection.

The smartphone users worldwide are not aware of the permissions as the basis of all malicious activities that could possibly operate in an android system and may steal personal and private information. Android operating system is an open system in which users are allowed to install application from any unsafe sites. However permission mechanism of and android system is not enough to guarantee the invulnerability of the application that can harm the user. In this paper, the permission scoring-based analysis that will scrutinized the installed permission and allows user to increase the efficiency of Android permission to inform user about the risk of the installed Android application, in this paper, the framework that would classify the level of sensitivity of the permission access by the application. The framework uses a formula that will calculate the sensitivity level of the permission and determine if the installed application is untrusted or not. Our result show that, in a collection of 26 untrusted application, the framework is able to correct and determine the application's behavior consistently and efficiently.

KEYWORDS

permission, permission scoring-based, malware Android phone, Security, Internet, malware.

1. INTRODUCTION

Nowadays, the advancement of technology is rapid. Today, a new product is being introduced to the market and in a week, month later a new one surfaces with a better functionality against its predecessor. Mobile phones are not exempted in the advancement of technology. From call and text only functionality, mobile phones became smartphones (Android) that serves as pocket computers. It is informing the entire user about the risk while using the application. In order to install the application from device you need the permissions that the application request. Most of the users are not paying attention or do not fully comprehend the requested permission. In addition, these permissions permit the malware to penetrate or exploit private data stored on device and perform malicious activities such as reading users private information, track user location, log-in credentials, and web browser history. Example of this permission is the INTERNET; many of the application communicate over the internet and malware developer advantage the use of this permission and combine with other permission [19].

In 2010, some of the Android developer Hans they just simply use it just to make sure that their application works properly. Therefore, a combination and unprofessional use of permission can take the advantage of stealing users' private data [10]. The existing security permission model of android has flaws that cannot protect the users' private data effectively. Several researchers, questioned the Android security model, and stated that the current permission model [11].

In (2015). The problems encountered by smartphone users in manipulating and maintaining the android malware security are the Absence of efficient Malware detector in Android Phones and Due to increasing numbers of Android Malware applications, a fast and reliable malware detector is necessary [13]. This proposal will conduct a study to detect the behavior of the malicious apps that can manipulate information on android devices. As a solution we present a permission-based scoring detection, which will evaluate the permission of the application and identify the application if its malicious or not The researcher achieve the process of detecting the malicious applications and develop an android application that can detect Malware applications in Android Phone, extract permissions of all installed android applications and evaluate the permissions that are extracted and determine malware application by the use of malware score formula.

2. REVIEW OF RELATED LITERATURE

A. PERMISSION ANALYSIS FOR ANDROID MALWARE (2015)

Detection. Once smart phones are infected with malware, users may face the following risks: disclosure of personal information, sent messages and read communications without permission, exploited the data with malicious intent. So the researchers PAMP, Permission Analysis for Android Malware Detection, which analyzes the Manifest file by understanding the Android Permission and by investigating malicious characteristics [1].

B .PERMISSION-BASED MALWARE DETECTION SYSTEM(2014)

PMDS System A cloud requested permissions as the main feature for detecting suspicious activities. PMDS applies a machine learning approach to categorize and determine automatically the harmful previously unseen application based on combination of permission required. In their study, they offer some discussion identifying the degree of android malware that can be detect and the prevention of malware by focusing on the permission they request. To understand the focus of the study, the set of permissions asked by the application corresponds to the behavior as either begin or malicious [4].

C.DREBIN: EFFECTIVE AND EXPLAINABLE DETECTION OF ANDROID MALWARE IN YOUR POCKET (2014).

"Malicious applications pose a threat to the security of Android malware." Researchers proposed DREBIN, method for detecting malware that enables identifying malicious application in Android by gathering as many features of an application as possible

D.THE POSSIBILITIES OF DETECTING MALICIOUS APPLICATIONS IN ANDROIDS PERMISSION (2013)

Study attempts to explore Collected relative large number of benign, malicious applications and conducted experiments and collected information based on the sample [3].

E.PUMA: Permission Usage To Detect Malware In Android (2013)

The presence of mobile devices has increased in our lives offering almost the same functionality as a personal computer. Android devices have appeared lately and, since then, the number of applications available for this operating system has increased exponentially. Google already has its Android Market where applications are offered and, as happens with every popular media, is prone to misuse. In fact, malware writers insert malicious applications into this market, but also among other alternative markets.” Researchers presented PUMA (Permission Usage to Detect Malware in Android), method for detecting malicious Android applications by analyzing the extracted permissions from the application itself.

F. Creating User Awareness Of Application Permissions In Mobile Systems.

Classifies the applications based on a set of custom rules if a rule is applied by the application it will mark as suspicious. Permission Watcher provides a home screen widget that aware users for potentially harmful applications. The methodology in this context relies on the comparison of the Android security permission of each application with a set of reference models for an application that manages sensitive data. The present researchers apply the idea of permission-based analysis to analyze the applications in order to know if the android app is malicious or benign.

G. Permission Watcher (2012)

The set of custom rules provides a home screen widget those aware users for potentially harmful application; the present researcher applies the idea of permission-based to track the behavior of the applications to know if the android app is malicious or benign [5]. Permission Flow tool that can easily identified. The system classified the application as benign.

H. Droidmat: Android Malware Detection Through Manifest And API Calls Tracing (2012)

The threat of Android malware is spreading rapidly, especially those repackaged Android malwares.” Presented DroidMat, a static feature-based mechanism to provide a static analyst paradigm for detecting the Android malware by extracting the information (Intents, permissions, etc.) from the application’s manifest and regards components (Activity, Receiver, Service) as points drilling down for tracing API Calls related to permissions.

3. Methodology And Design

Conceptual Framework

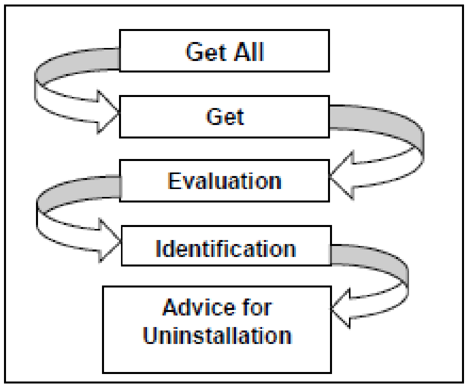


Fig. 1 Malcure Conceptual Design

This section will present the overview of the Malcure framework and the description of each phase. System frameworks illustrate the flow of each phase in working out to analyze the application during the scanning.

3.1 MALCURE

Will scan for the apps that may contain malwares that could leak sensitive information. Just after the scan button was tapped, each of the apps will processed, so that each of the app's permissions will be directly extracted, and therefore will undergo permission based scoring. The permission scoring analysis will be performed to check if the permission score has exceeded the malicious standard score or not. If yes, the application will be advised for uninstallation.

3.2 GET ALL APPLICATIONS

The process where all the applications will be process to be prepared for extraction of the permissions.

3.3 GET PERMISSIONS

The app's permissions are directly extract from the application, and there is no need for DE compilation of the base file.

3.4 EVALUATION

This is where all the permissions are evaluated based on the scores set on the sensitivity of a permission ranging from 1 to 6, making 1 as the Neutral permission, and 2 to 6 are the sensitive permissions, and all are processed base on a formula.

3.5 IDENTIFICATION

The overall malicious score is determined in this phase, and therefore will be advice for un installation if the score exceeds the malicious standard score.

3.6 ADVICE FOR UNINSTALLATION

When a particular application is judge as malicious, Malcure will open a window, where the app is advised for uninstallation

3.7 APP SCANNING FRAMEWORK

The process of Malcure scanning mechanism, at the start of this function, there will be scanning performed in a loop of user-defined and system applications that directly extracts each of the application permissions to be evaluated and process with the Permission Score Analysis and the Formula to determine the Malicious Score of a particular application. Once an application has exceeded that malicious standard score, its advice for uninstallation

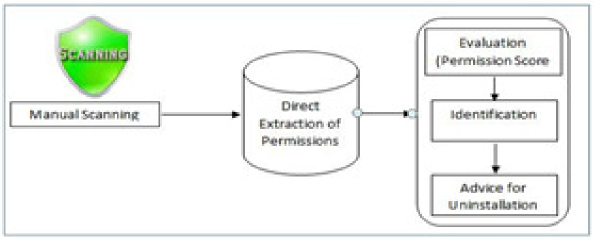

SCANNING
Manual Scanning
Direct Extraction of Permissions
Evaluation (Permission Score
Identification
Advice for Uninstallation

Fig.2 Malcure App Scanning Framework

In this section, we will briefly discuss the permissions and their sensitivity and malicious scores that will determine the capability of an app in stealing sensitive information. In addition, the table that represents the sensitive permissions and their malicious scores of a particular application. Once an application has exceeded that malicious standard score, it is then advised for uninstallation.

3.8 Permissions Sensitivity and Their Malicious Score

Malicious Strings	Sensitivity	Malicious Score
READ_SMS	MODERATE HIGH	3
SEND_SMS	HIGH	5
RECEIVE_SMS	HIGH	5
WRITE_SMS	HIGH	5
PROCESS_OUTGOING CALLS	VERY HIGH	6
MOUNT_UNMOUNT_FILE SYSTEMS	MODERATE	2
READ_HISTORY_BOOKMARKS	MEDIUM HIGH	4
WRITE_HISTORY_BOOKMARKS	MODERATE HIGH	3
READ_LOGS	VERY HIGH	6
INSTALL_PACKAGES	VERY HIGH	6
READ_PHONE_STATE	MODERATE HIGH	3
READ_CONTACTS	MEDIUM HIGH	4
ACCESS_FINE_LOCATION	MODERATE HIGH	3

Table 1 Permission Sensitivity and their Malicious Score

The figure shows the formula where R, is the Overall Malicious Score. M, which is the total scores of the sensitive permissions. C is the number of Neutral or Benign Permissions

3.9 Untrusted Score Evaluation

Malicious Strings	Sensitivity	Malicious Score
RECEIVE_BOOT_COMPLETED	NEUTRAL	1
SEND_SMS	HIGH	5
READ_PHONE_STATE	MODERATE HIGH	3
READ_LOGS	VERY HIGH	6

Table2. Sample Application with Permissions

$$R \quad \frac{14}{14+1}$$

$$= 0.93333 \text{ Overall}$$

Figure 4. Sample Result

Shows a sample application with the following permissions. Now, using the formula we will get: Figure 4. Sample Result

Figure shows the result from Table 2, which is considered to be an untrusted because of the fact that it exceeded the untrusted standard score which 0.70.

3.1.1 Untrusted Standard Score

Come up with 0.70 untrusted app standard score, based on multiple mock up tests and analyzations on multiple untrusted applications, and discovered that even on applications that has only two permissions. The other is neutral and the other permission is sensitive with 2 points, it will be considered an untrusted, which is an appropriate action for anti-malware application. Any application with overall untrusted score equal or more than to 0.70 will be considered an untrusted.

The Untrusted Standard Score Basis

Device Model with Built in Apps only	**Highest Score**	**Package Name**
Cherry Mobile Flare	0.69	Com.cherryplay
Acer Liquid z160	0.66	Com.backuptester
Samsung Duos	0.69	Com.hangouts
Myphone Rio	0.67	Com.facebook.orca
Sony Erikkson Curve	0.68	Com.backuptester

Total = 3.39 / 5 → Average = 0.68 rounded up to 0.70

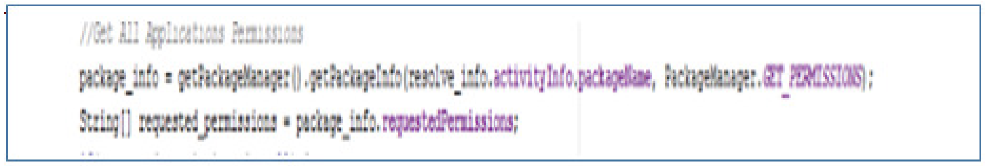
Table3. Untrusted Standard Score Basis

Table shows the basis of the untrusted detector Standard Score is by sampling some smartphones with different brands, stored with only built only applications and we've calculated the highest scores of each smartphones and get there average. Because of the fact that smartphone manufacturers do not develop built in applications with malwares, every time a user application exceeds that score, it also exceeds the basis of the manufacturer in developing clean applications. Failure to do so will result to disclosure of the license to produce Smartphones with Android OS. This standard is our basis that every time an application exceeds that standard, our study and developed system will consider it a Malware.

4. Results And Discussion

4.1 Permission Extraction

Our way of extracting the permissions of every application was successful because of the fact that Android has a predefined class of directly extracting every permission without decompiling the APK base file.



```
//Get All Applications Permissions
package_info = getPackageManager().getPackageInfo(resolve_info.activityInfo.packageName, PackageManager.GET_PERMISSIONS);
String[] requested_permissions = package_info.requestedPermissions;
```

Figure 6. Permission Extraction

The above line of codes represents the extraction of all the permissions that comes from the application, whether it is from the system or the user.

4.2 Untrusted Detection

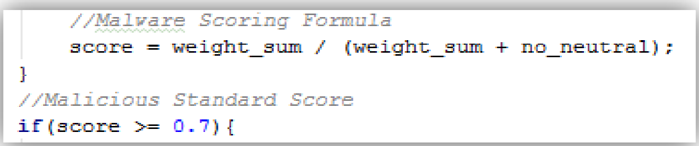
This is the process which shows on how a Malware is detected, through Evaluating the permissions extracted based on the thirteen sensitive permissions, then using the malicious scoring formula, which then states if the application is advisable for uninstallation or not.

4.3 Permission Validation

By comparing all of the permissions of a particular application to the sensitive permission stated in Table 2, we were able to come up with the malicious score that are necessary for coming up with the overall malicious score. Once a permission matches with the sensitive permissions.The score matching the sensitivity of the permission is incremented, and all remaining permissions which did not match, will be considered as neutral permissions.

4.4 Malware Scoring Formula/Untrusted Detection

Come up with untrusted scoring formula that was based on a study that we slightly modified, due to reasons that we want untrusted to be fast and efficient, because on its original study, it included process and third party resources that causes the overall process to be slow, and comprise large memory. The malware standard score on the other hand was the result of multiple mock up tests and analyzations on multiple malware applications, and we've discovered that even on applications that has only 2 permissions. The other is neutral and the other permission is sensitive with 2 points, it will be considered a malware, which is an appropriate action for anti-malware application. Any application with overall malware score equal or more than to 0.70 will be considered a malware.

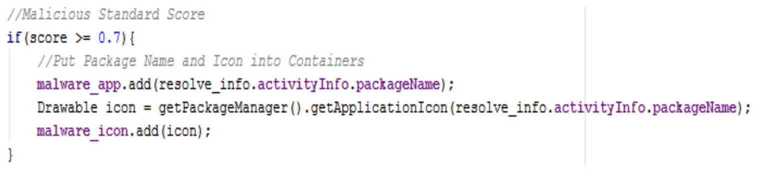


```
    //Malware Scoring Formula
    score = weight_sum / (weight_sum + no_neutral);
}
//Malicious Standard Score
if(score >= 0.7){
```

Figure 7 Untrusted Scoring Formula/Untrusted Detection
Shows the Untrusted Scoring Formula and Untrusted Standard Score.

4.5 Uninstall Recommendation

After the processing of all the scores of the matched sensitive permissions, a final and overall malicious score is generated using the formula, then a condition is formulated that when the overall malware score is equal or more than the malware standard score, that particular application will be advised for uninstallation with the consent of the user.



```
//Malicious Standard Score
if(score >= 0.7){
    //Put Package Name and Icon into Containers
    malware_app.add(resolve_info.activityInfo.packageName);
    Drawable icon = getPackageManager().getApplicationIcon(resolve_info.activityInfo.packageName);
    malware_icon.add(icon);
}
```

Figure 8 Uninstall Recommendation

4.6 Graphical User Interface

This represents the interaction between the user and Untrusted and how the user can manipulate untrusted, from scanning to determining if the application is a malware or not, with its following process:

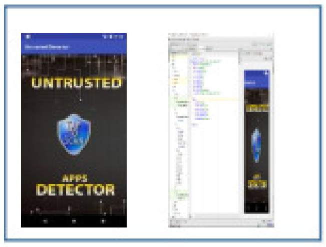
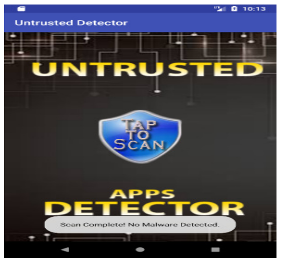


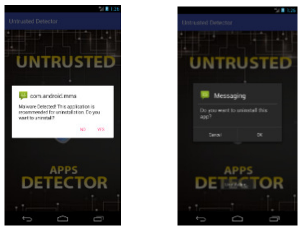
Figure 9. Tap to Scan

Whenever the user taps the shield icon, Untrusted immediately starts its scanning from the applications from the user and the system, and therefore starts the process from validation, evaluation, identification and advice for uninstallation.



Scan Complete! No Malware Detected.

Figure 10. No Malware Detected



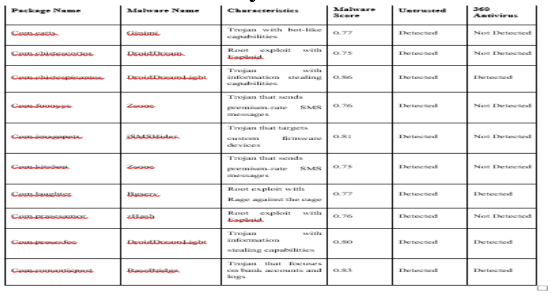
Malware Detected! This application is recommended for uninstallation. Do you want to uninstall?

Do you want to uninstall this app?

Figure 11. Untrusted Detected

Shows when the malware score is equal or more than the malware standard score, and therefore detects a malware, a dialog box appears advising the user to uninstall the particular application, and then after that another dialog box appears clarifying the user's decision. We do this, because we observe the full right of the user to keep the application if the user wanted to. But when the user accepts, the application is uninstalled immediately. There are multiple applications detected as malware, the event that will happen is that after the first app has been take cared for, a next dialog box pertaining to the next malware detected will appear.

4.7 Untrusted vs. 360 Antivirus Security First Version



Package Name	Malware Name	Characteristics	Malware Score	Untrusted	360 Antivirus
Com.catts	Gininmi	Trojan with bot-like capabilities	0.77	Detected	Not Detected
Com.chistescortos	DroidDream	Root exploit with Exploid	0.75	Detected	Not Detected
Com.chistespicantes	DroidDreamLight	Trojan with information stealing capabilities	0.86	Detected	Detected
Com.fooyys	Zsone	Trojan that sends premium-rate SMS messages	0.76	Detected	Not Detected
Com.imagepets	jSMSHider	Trojan that targets custom firmware devices	0.81	Detected	Not Detected
Com.kitchen	Zsone	Trojan that sends premium-rate SMS messages	0.75	Detected	Not Detected
Com.laughter	Basery	Root exploit with Rage against the cage	0.77	Detected	Detected
Com.passcounter	zHash	Root exploit with Exploid	0.76	Detected	Not Detected
Com.prasesfee	DroidDreamLight	Trojan with information stealing capabilities	0.80	Detected	Detected
Com.romanticpast	BaseBridge	Trojan that focuses on bank accounts and logs	0.83	Detected	Detected

Table 3. Untrusted vs. Other Permission Based Malware Detector

Shows that we have taken 10 updated sample malwares that are likely used to attack smartphone devices and steal personal information. Out of 10 sample malware tested, 360 antivirus just detected 4, while Untrusted perfectly detected all of them. So, we conclude that Untrusted is more reliable than the first version of the 360 security, and with more improvement and development, it will be a remarkablemalware detector for Android Operating system, with the main feature of fast, memory friendly and reliability.

Surveys

Disguised 10 malicious software and invited 10 individuals to try Untrusted and compare it with the first version of 360 Antivirus, with the disguised software installed, and ask their opinions and statements about the differences between the malware detector, and which is faster and more reliable for them in detecting malwares.

Survey Result

Based on the data collected from 10 participants, comprised of average users, techy geeks and researchers, come up with this graphical representation that helps us conclude on the performance, reliability, memory friendliness and usage preferability of Untrusted.

Chart Presentation

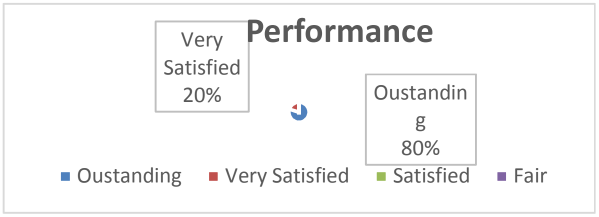


Figure 12. Performance Survey Result

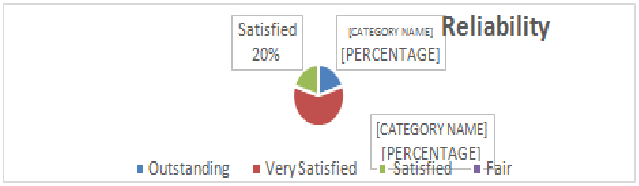


Figure 15. Reliablity survey Result

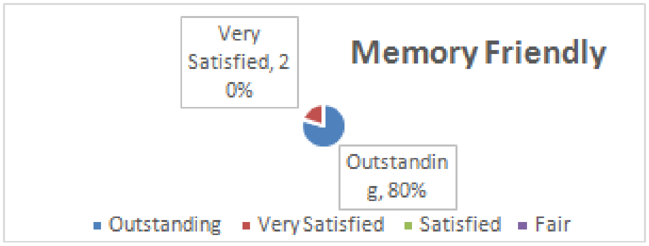


Figure 16. Memory Friendliness Survey Result

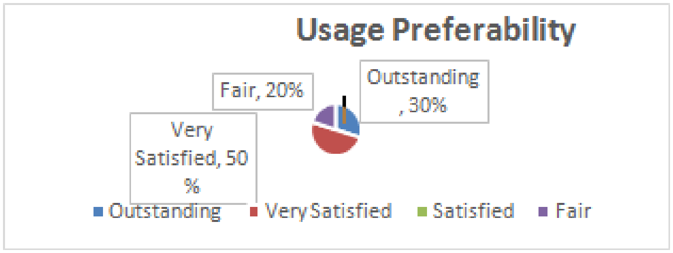


Figure 17. Usage Preferability

SURVEY CONCLUSION

Based on the survey that conducted on 10 Smartphone users, we're able to collect data that helps us prove that Untrusted is fast, reliable, memory friendly and users are going to use it. With 95% on approval on Performance, 75% percent on Reliability, 95% on Memory Friendliness and 80% percent on Usage Preferability, our study and all of its methodology are proven base on the user's experience on the developed system.

5. SUMMARY AND CONCLUSION

SUMMARY

Android Untrusted is considered as one of the problem that many android users encountered. The proposed untrusted detection for android phones that will identify the malicious application that is installed on the device. Based on the experiment that we conducted it shows that untrusted is effective in detecting malicious application. Untrusted detection was effective and efficient in extracting the permission without decompiling the apk. To get the following permission use getPackageManer().getPackageInfo() to extract the permission. The researcher observed that by using the package manager it's achieve the process of extracting the permission much faster. It's also see that permission-scoring formula is effective for evaluating the level of permissions in order to decide if the application is malicious or benign.

Based on the experiment and survey that the researcher conducted it shows that untrusted is effective in capturing the malware application. It shows that the untrusted application that installed on the android device was captured by the untrusted detection.

6. CONCLUSION

The UN system is effective in providing a solution by detecting the malicious application that can penetrate the android device. The researcher presented a methodology and architecture for measuring the permission accessed by the application using permission-scoring formula, which will identify if the application was manifested with malicious permission. Using the permission scoring detection, and it's satisfies the Untrusted Detection objectives to capture the malware application. Using this anti-malware application, android user will be aware of the applications and its true behaviors.

REFERENCES

[1] NguyenVietD.etal. (2015) " PermissionAnalysisforAndroidMalwareDetection".Retrievedfromhttps://www.researchgate.net/profile/Pham_Giang4/publication/296704790_Permission_Analysis_for_AndroidMalware_Detection/links/56d9bce708aee1aa5f8291f4.pdf

[2] Isohara, T., Takemori, et. al. (2011)."Kernel-basedBehavior Analysis for Android Malware Detection".Retrieved from http://ieeexplore.ieee.org/abstract/document/6128277

[3] Huang, C. Ts ai, et. al. (2013). "The PerformanceEvaluation on Permission- Based Detection for Android Malware". Retrieved fromhttps://www.link.springer.comchapter10.1007/978-3-642-35473-112

[4] Rovelli, P., & Vigfússon, Ý . (2014).PMDS:"Permission-Based Malware Detection System". Retrievedfrom https://www.link.springer.com/chapter10.1007/978-3-319-13841-1_19

[5] Struse, E., Seifert, J., Üllenbeck, S.,Rukzio, E., & Wolf, (2012). "PermissionWatcher: Creating User Awarenessof Application Permissions in Mobile Systems". Retrievedfrom https://www.link.springer.com/chapter10.1007/978-3-642-34898-3_5

[6] Sbirlea, D. Burke, et. al (2013). "Automatic Detection ofInter-Application Permission Leaks in Android applications".Retrieved from http://www.ieeexplore.ieee.org/abstract/document/6665098/

[7] Wu,D.et.al.,(2012). "DroidMat: API Calls Tracing".Retrieved from http://www.ieeexplore.ieee.org/abstract/document6298136

[8] Sanz, B., Santos, et. al. (2013). "PUMA: PermissionUsage to Detect Malware in Android". Retrieved fromhttps://link.springer.com/chapter/10.1007/978-3-642-33018-6_30

[9] Arp, D., et. al. (2014). Drebin: "Effective andExplainable Detection of Android Malware in Your Pocket" Retrieved from https://www.researchgate.netprofile/264785935_DREBIN_Effective_and_Explainable_Detection_of_Android_Malware_in_Your_Pocket/links/53efd0020cf 26b9b7dcdf395.pdf

[10] Barrera, D., Kayacik, et. al., (2010). "Methodology forempirical analysis of permission-based security models and its application to android categoriesand subject description. In processing of 17th ACm conference on computer and communication security New York" NY, USA ACM 2010. P 73-74 http://dx.doi.org/10.1145/1866307.18663317

[11] Enck W.Gilbert, et. al., (2014). "TaintDroid:An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones", Retrieved fromhttp://www.dl.acm.org/citation.cfmid2619091

[12] K.Mathur et.al., "A Survey on Techniques in Detection and Analyzing Malware Executables, Int. Journal of Advanced Research in Computer Science and Software Engg.", India, vol. 3, issue 4, pp. 422- 428, 2013.

[13] R.Sato, et.al., "Detecting Android Malware by Analyzing Manifest Files", pp. 2331, 2013.Android Permissions Demystified. [Online] Available: https://www.truststc.org/pubs/848.html. [Accessed: 06-Nov-2015].

[14] P.Faruki, V.Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, Andro Similar: "Robust Statistical Feature Signature for Android Malware Detection, Proc. 6th Int. Conf. Secur. Inf. Networks", pp. 152 159, 2013

[15] Mengyu Qiao, et. al., "Merging Permission and API Features for Android Malware Detection", vol. 00, no., pp. 566-571, 2016, doi:10.1109/IIAI-AAI.2016.237

[16] K.Xu, Y. Li, and R. Deng, ICCDetector: "ICC-Based Malware Detection 4 on Android, in Proc. of IEEE Transaction in Information Forensics and security", vol. 11, no. 6, June 06, 2016

[17] R. Raveendranath, V. Rajamani, A. J. Babu, and S. K. Datta, “Android malware attacks and countermeasures: Current and future directions, 2014 Int. Conf. Control. Instrumentation Commune. Compute”. Technol., pp. 137143, 2014

[18] R. Johnson, C. Gagnon, Z. Wang, and A. Stavrou, “Analysis of Android Appss Permissions, in Proc. of 6th IEEE Int. Conference of Software Security and Reliability Companion, Maryland”, pp. 45-46, 2012

[19] “Android applications ... and more (ninja!) - GoogleProject Hosting”.[Online].Available: https://code.google.com/p/androguard/. [Accessed: 01- Dec 2015].

[20] L.Wenjia, D. Guqian, ““An SVM Based approach”, in Proc. of IEEE 2nd Int. Conference on Cyber Security and Cloud Computing”, 2015.

[21] Android developer guide permission 9(WWWdocument). Google. URL, http;//developer.android.com/guide/topics/manifest/permission-element.html#package; 2014[accessed 2512.14]

[22] Wu D J, Mao C H, Wei T E, et al. Droidmat: Android malware detection through manifest and API calls tracing. In: Proceedings of the 7th Asia Joint Conference on Information Security (Asia JCIS). Piscataway: IEEE Press, 2012. 62–69

[23] Zhou Y J, Wang Z, Zhou W, et al. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets. In: Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, 2012. 25: 50–52

[24] Bläsing T, Batyuk L, Schmidt A D, et al. An android application sandbox system for suspicious software detection. In: Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE). Piscataway: IEEE Press, 2010. 55–62

[25] Stevens R, Ganz J, Filkov V, et al. Asking for (and about) permissions used by android apps. In: Proceedings of the 10th Working Conference on Mining Software Repositories. Piscataway: IEEE Press, 2013. 31–40

[26] Karim M Y, Kagdi H, Di Penta M. Mining android apps to recommend permissions. In: Proceedings of the 23th IEEE/ACM International Conference on Software Analysis, Evolution, and Reengineering. Piscataway: IEEE Press, 2016. 427–437

[27] M. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic detection of capability leaks in stock Android smartphones. In Proceedings of the 19th Network and Distributed System Security Symposium (NDSS), Feb. 2012.

[28] Kim, J. I. Cho, H. W. Myeong, and D. H. Lee. A study on static analysis model of mobile application for privacy protection. In J. J. (Jong Hyuk) Park, H.-C. Chao, M. S. Obaidat, and J. Kim, editors, Computer Science and Convergence, volume 114 of Lecture Notes in Electrical Engineering, pages 529-540. Springer Netherlands, 2012. 10.1007/978-94-007-2792-2 50.

[29] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In Proceedings of the 18th ACM conference on Computer and communications security, CCS '11, pages 627-638, New York, NY, USA, 2011. ACM.

[30] Sina science and technology, http://tech.sina.com.cn/it/20130510/07478325514.shtml[EB/OL].May 2013.

[31] DoNews.http://www.donews.com/net/201305/1495781. shtm[EB/OL]. May 2013.

AUTHOR

Engr. Jackelou S. Mapa, MIT Information Technology Education Program Head Saint Joseph Institute of Technology Montilla Boulevard, Butuan City, Philippines