International Journal of Security and Its Applications
Vol.7, No.5 (2013), pp.121-130
http://dx.doi.org/10.14257/ijsia.2013.7.5.10

# A Novel Method to Avoid Malicious Applications on Android

Sangho Lee and Da Young Ju

*Yonsei University*
*sangholee@yonsei.ac.kr, dyju@yonsei.ac.kr*

***Abstract***

*Security threats on modern smartphones are increased rapidly with the excessive numbers of the users. According to the reports, it is exposed that most security issues are introduced from usage of malicious applications. Operating systems with the permission model, such as Android, accept us to install third-party applications. This leads attackers enable to inject exploits into a clean application, so that an application apparently works as normal but executes malicious functions in the background. Therefore, we introduce with analysis of the method to prevent an installation of malicious applications using permissions using Maximum Severity Rating (MSR) classification. Then, the method to enhance an ability to perceive warning signs in the procedure of an application installation and comparison of its effectiveness with existing method is introduced. Overall, the processes assist the users to make a better decision with detailed information.*

***Keywords:*** *Android applications, malicious application assessment, enhanced user interfaces*

## 1. Introduction

Smartphone OS accepts the users to install third party applications. Based on unexpected population of third party application usage, Smartphone users are rapidly growing. According to the report by Gartner in 3rd quarter in 2013[15], the users of Android are 72.4%, and thus dominate Smartphone market. While Android is widely spreading among Smartphone users, its security threats are also increasing immensely. The report created by McAfee claims that latest exploits are mostly distributed via malicious applications which pretend to be safe [16]. Malicious applications can only execute malfunctions with previously granted permissions. For this reason, attackers require more permissions than what the normal applications need. Serious problems, such as accessing call log, SMS and contact information can be caused if an infected application is running. Therefore, in this paper, the proposed methods focus especially on the procedure of permission agreement, and concentrate to help the users verifying an application properly with rejection of a malicious application as an extensive work from the previous paper [17]. To implement this, this paper obtains the process to examine an application with proposed method with analysis of permission-grant model [4, 6, 7] which Android employees. Also, enhanced visualization of permission set is proposed to display information effectively with comparison of previous user interface to provide the better attractions. Throughout this, we expect to replace the previous permission-grant model which includes lower reliability with text format [9] to the proposed model with guarantee of an integrity and better user interface. To achieve this, relative works are mentioned to claim the differences with previous works. Then, calculate a Maximum Severity Rating (MSR) to judge if an application is malicious using a proposed method. Enhanced visualization of permission set is provided to improve a visual accessibility with increment of the right

ISSN: 1738-9976 IJSIA

decisions by the users. The processes are described in Figure 1. Finally, a conclusion reviews proposed methods with limitations and necessary works in advance.

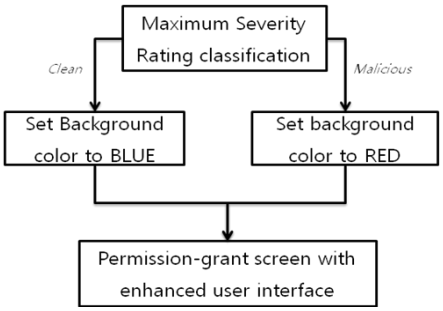


**Figure 1. Brief Explanation of the Overall Process Proposed in this Paper**

## 2. Related Works

Because of difficulties to recognize an actual security vulnerability level by watching only permissions at an application installation stage, various prior researches are performed. Research [3, 4, 14] present the methods to classify malicious applications by searching APIs which are called in an application. They create a list which can be performed as malicious functions. The list can be pre-generated by human-classification. Then, examination for existing application is performed. Once code segment of an application calls a malicious function, they classify it as a malicious application. However, these approaches are not adaptable in an actual android system alone since these require a modification of the kernel, which is difficult. Also, it is very important to make the method guarantee to classify accurately because a final decision which machine make must be 100% accurate in an active prevention of malicious app installation. Yet, these are not guaranteed integrity of their examination result. Similarly, [5] offers a package to detect malicious operation, calls Aurasium. It creates a bridge to enable redirection of important kernel functions for malicious application classification to its own function. As long as functions calls in an application are automatically triggered into Aurasium, a malicious operation from an application is filtered in the package. Nevertheless, due to the possibilities to detect a proper application as a malicious, final decision should be made by the user. The guarantee of 100% accuracy is impossible. Our method focuses not on improving an algorithm by implementing automatic permission authorization system for users to make a better decision, but on maximizing the user's attention and providing the useful information regarding a permission of an application to install. To complete our method, Maximum Severity Rating (MSR) classification with an analysis is explained first. Using this, enhancement of visibility and comparison of previous interface is provided. As the result, these methods aggregate sequentially into the installation procedure to help the users to make a better recognition from the users.

## 3. Maximum Severity Rating (MSR) Classification

Most Smartphone users are not technically trained. They have limited abilities to see whether an application is malicious or clean although they try to reject an installation of malicious applications. To assist users, Maximum Severity Rating (MSR) classification attempts to find malicious applications by examining requested permissions. Our interface is designed to display the result of classification with alarming colors to let the users to easily

 

perceive the risks. Since the proposed method does not always guarantee to produce accurate results, the final decision to install an application is to be made by a user. In other words, the proposed method only assists the users who do not have sufficient knowledge about a permission-based security model.

### A. Methodology

Maximum Severity Rating (MSR) in this paper is classified by the permissions. Developers have to specify required permissions in manifest file as specified in Figure 2, in order to acquire an access to each of the functionalities.

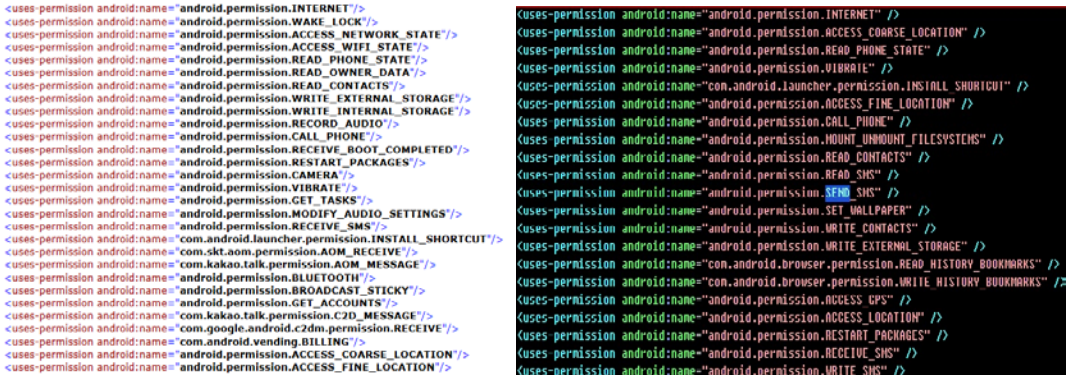


**Figure 2. Requested Permissions Comparison - KaKaoTalk(Left, Clean) and Monkey Jump 2(Right, Malicious)**

However, because malicious operations require permissions which normal application does not necessary. In this paper, statistical analysis result is employed to calculate MSR. According to the latest research [1], there are 20 permissions which are relative to the classification of a malicious application. And also, there are 20 permissions which are acquired by a normal application. Some of them are similar in the ranking, for example of the permission INTERNET but malicious applications demand unnecessary permissions, such as READ_HISTORY_BOOKMARKS. In this paper, MSR classification calculates the permissions set of an application to inform whether it is malicious or not to the user. Along with classification, an analysis of parsed permissions and the weight value is performed. To initiate our method, the method combines statistical analysis result of malicious and normal applications [1]. Their percentages are indicated as a weight value. Because an application trying to install can be different from the results of statistical analysis, the permissions which are not used in one side are fulfilled by 1, not 0[2]. Then, normalization process is achieved to judge in the same start point. The permissions are normalized by INTERNET, which sets to 100%. An equation for normalization is as follows:

$$MN_i = \frac{100}{M_1} M_i \quad , 1 \le i \le 25$$

$$NN_i = \frac{100}{N_1} N_i \quad , 1 \le i \le 25$$

**Table 1. Count of Requested Permissions (%) in Malicious (M) and Clean (N) Application and Normalized (MN/NN) Vector Table**

| | | M(%) | N(%) | MN(%) | NN(%) |
|---|---|---|---|---|---|
| 1 | INTERNET | 96 | 85 | 100 | 100 |
| 2 | READ_PHONE_STATE | 84 | 34 | 88 | 40 |
| 3 | ACCESS_NETWORK_STATE | 75 | 56 | 78 | 66 |
| 4 | READ_HISTORY_BOOKMARKS | 58 | 1 | 60 | 1 |
| 5 | WRITE_HISTORY_BOOKMARKS | 57 | 1 | 59 | 1 |
| 6 | ACCESS_WIFI_STATE | 54 | 10 | 56 | 12 |
| 7 | WRITE_EXTERNAL_STORAGE | 53 | 36 | 55 | 42 |
| 8 | VIBRATE | 53 | 20 | 55 | 24 |
| 9 | RECEIVE_BOOT_COMPLETED | 47 | 9 | 49 | 11 |
| 10 | ACCESS_COARSE_LOCATION | 45 | 25 | 47 | 29 |
| 11 | ACCESS_FINE_LOCATION | 44 | 26 | 46 | 31 |
| 12 | WAKE_LOCK | 37 | 12 | 39 | 14 |
| 13 | READ_CONTACTS | 29 | 8 | 30 | 9 |
| 14 | SEND_SMS | 27 | 3 | 28 | 4 |
| 15 | SET_WALLPAPER | 27 | 5 | 28 | 6 |
| 16 | READ_SMS | 22 | 1 | 23 | 1 |
| 17 | ACCESS_LOCATION_EXTRA_COMMANDS | 22 | 1 | 23 | 1 |
| 18 | WRITE_SETTINGS | 19 | 4 | 20 | 5 |
| 19 | CALL_PHONES | 19 | 11 | 20 | 13 |
| 20 | RECEIVE_SMS | 17 | 1 | 18 | 1 |
| 21 | CAMERA | 1 | 7 | 1 | 8 |
| 22 | GET_TASKS | 1 | 4 | 1 | 5 |
| 23 | READ_CALANDER | 1 | 4 | 1 | 5 |
| 24 | WRITE_CALANDER | 1 | 4 | 1 | 5 |
| 25 | GET_ACCOUNTS | 1 | 3 | 1 | 4 |

To implement this, a statistic table of malicious and clean applications is allocated to the vector table. Then, requested permissions set in an application is generated to another vector table. The permissions not specified in statistic table will be discarded. For each corresponding index, store the value of 1 if the permission is requested. Otherwise, store the value of 0. Normalized weight $MN_i$ and $NN_i$ are multiplied by the vector set of requested permissions. Sequentially, an average weight value is calculated as follows:

$$Avg_{xMR} = \frac{rMR\sum_{i=1}^{25} xMR_i}{25}$$ , rMR is sample difference ratio

$$Avg_{xNR} = \frac{rNR\sum_{i=1}^{25} xNR_i}{25}$$ , rNR is sample difference ratio

Once average value is calculated for the probabilities of malicious and normal applications, comparison of those two values are performed. An application is indicated as a malicious application if the value of normal application is less than malicious application. In contrast, it is recognized as a clean application.

### B. Test Results

To execute the test to compare the result of Maximum Several Rating classification, KaKaoTalk published by KaKao Inc.[11] and Angry Birds published by Rovio [10] is used for the case of a clean application, and Monkey Jump 2 [12] is assessed as the case of a malicious application. The measurements result shown in Table 2 shows that the risk value of a clean application (NNR, ANR) exceeds the weight of a malicious application (NMR, AMR)

 

when an application doesn't include a malicious function. Meanwhile, a ratio of a malicious application is higher than a clean since an application consists of an exploits. The reason of variation of measurements result comes from an effectiveness of permissions which is only requested in a malicious application, such as READ_HISTORY_BOOKMARKS and WRITE_HISTORY_BOOKMARKS. Because their weight ( $MW_4$ , $MW_5$ ) is sufficiently dominant compared to the value of the malicious application ( $NW_4$ , $NW_5$ ), the test result may be larger than the value of the clean one.

**Table 2. Malicious Ratio (XMR), Clean Application Ratio(XNR), and Permission Vector (XP) of KakaoTalk (N), Monkey Jump 2 (M), and Angrybird (A)**

| | NP | NMR | NNR | MP | MMR | MNR | AP | AMR | ANR |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 100 | 100 | 1 | 100 | 100 | 1 | 100 | 100 |
| 2 | 1 | 88 | 40 | 1 | 88 | 40 | 1 | 88 | 40 |
| 3 | 1 | 78 | 66 | 0 | 0 | 0 | 1 | 78 | 66 |
| 4 | 0 | 0 | 0 | 1 | 60 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 59 | 1 | 0 | 0 | 0 |
| 6 | 1 | 56 | 12 | 0 | 0 | 0 | 1 | 56 | 12 |
| 7 | 1 | 55 | 42 | 1 | 55 | 42 | 0 | 0 | 0 |
| 8 | 1 | 55 | 24 | 1 | 55 | 24 | 0 | 0 | 0 |
| 9 | 1 | 49 | 11 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 1 | 47 | 29 | 1 | 47 | 29 | 1 | 47 | 29 |
| 11 | 1 | 46 | 31 | 1 | 46 | 31 | 0 | 0 | 0 |
| 12 | 1 | 39 | 14 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 1 | 30 | 9 | 1 | 30 | 9 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 1 | 28 | 4 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 1 | 28 | 6 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 1 | 23 | 1 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 1 | 20 | 13 | 1 | 20 | 13 | 0 | 0 | 0 |
| 20 | 1 | 18 | 1 | 1 | 18 | 1 | 0 | 0 | 0 |
| 21 | 1 | 1 | 8 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 1 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 1 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Avg. | | 27 | 33 | | 26 | 24 | | 15 | 20 |

The result demonstrates that the permission INTERNET and ACCESS_NETWORK_STATE are requested in almost applications including malicious ones, so that meaningless permissions are discarded. Moreover, it represents that category is an important factor to decide the scale of the calculation result, even in the same class. For example, the value of the game, Angry Birds, is significantly less than the messenger, KaKaoTalk. According to an analysis, this variation comes from the characteristic of the functions. In detail, Angry Birds does not need functionalities which KaKaoTalk requires, such as reading a message, contact, and location information. However, a scale difference does not affect a final decision because a weight value is equally not summed up during the process of multiplication if an application doesn't request certain permissions. For this reason, once you calculate with the same formula, a condition itself, which is the rule that an application is recognized as malicious if xMR is greater than xNR, does not influence a decision-making process.

## 4. Enhanced Visualization of Permission Set

Proposed method, MSR classification, averages risk weight of requested permissions to determine whether an application is malicious or clean. However, there is possibility to make

a fault decision although the test result of above method describes the great performance to classify a malicious application. Incorrect decision leads to serious problems, such as installing a malicious application or declining to install a clean application based on a fault decision. To protect unexpected behaviors, MSR classification must be applied to assist a correct decision, and an installation must be processed with an agreement from the user. Since enhanced visualization based on improved user interface can lead the users more cognitive to the risks [13], this paper proposes a user interface of permission agreement screen which is redesigned for enhancement of the ratio to make the right decision.

### A. The Problem of an Existing Permission-grant Screen

When you install an application, Figure 3 is displayed. In this screen, the installation manager requests an agreement to the user to grant permissions. After you press install button to accept to install, an application is assumed that the user acknowledges the permissions described in the screen.

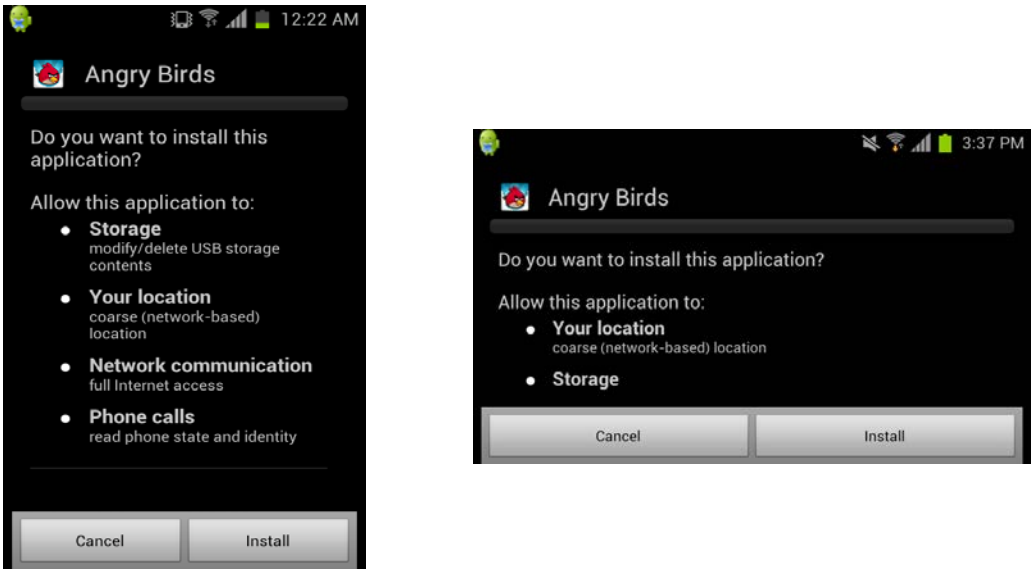


**Figure 3. Existing User Interfaces of an Application Installation - vertical(up) and Horizontal(down)**

Therefore, the screen should explain about possible security threats relative to consent of permissions. However, Android OS doesn't provide such descriptions. The problems of existing permission-grant model in the process of an installation based on Android OS are as follows – First, existing model doesn't provide a heuristic analysis report to assist the users' decision, whether an application includes a malicious functions or not. Because an assumption that the users are not active in reading a technical message is important for the general users who are not familiar with Android system, the solution which provides a comprehensive assessment of requested permissions is crucial to improve the users' right decision. Second, existing system only explains details of permissions which an application is currently trying to get an agreement and doesn't provide risks of an independent permission compared to entire permissions. As of a permission INTERNET, which is one of the most dangerous permission, it is not risky in practice since this permission is used in most applications which requires accessing an internet. Therefore, an analysis of an independent

 

permission must be proceed to help the users to examine whether an application is dangerous or not, and this makes the users efficient to recognize a potential risks. To dispatch above problems, this paper proposes the design which maximizes the users' awareness of an application installation.

### B. Proposed Permission-grant Screen

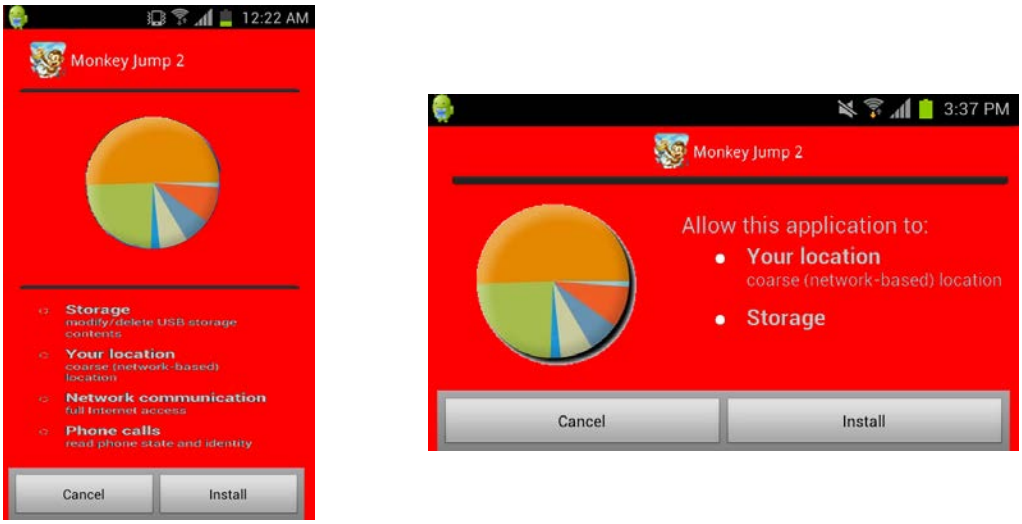


**Figure 4. Improved User Interface with an Example of a Malicious Application – Vertical(left) and Horizontal(right)**

Since existing permission-grant screen cannot aware the users in the case of malicious application, enhanced user interface with maximization of visibility is introduced to transmit the messages to the users more effectively. In order to supplement, the proposed design employees a comprehensive assessment to examine a malicious application which is not provided in an existing permission-grant screen. Background color of the screen is changed to blue if an analysis result is reported as clean. Otherwise, Red is displayed on the background screen with the result of malicious, as described in Figure 4. A graph is used to present the potential risks of each permissions. To calculate this, Degree of risk (DOR) is proposed. MSR classification result is needed to establish an equation of DOR. If the result value is indicated as malicious, DOR is calculated using below equation.

$$DOR_i = \frac{MN_i}{\sum_{j=1}^{25} MN_j P_j} 100, \quad 1 \le i \le 25$$

Likewise, the following formula is used if an application is reported as clean.

$$DOR_i = \frac{NN_i}{\sum_{j=1}^{25} NN_j P_j} 100, \quad 1 \le i \le 25$$

In above equations, DOR is expressed in percent. MN is normalized frequency of malicious applications usage shown in Table 1, and NN is a normalized count of the safe applications. P is a permission vector of an appropriate application, expressed as 0 or 1.

Calculated DOR is represented as the background color, as described in Figure 4 if an application is malicious.

## 5. Conclusion

In this paper, we propose and analyze the method to examine a malicious application using Maximum Severity Rating(MSR) classification. In addition, enhanced user interface with comparison to an existing screen is introduced with redesign of a permission-grant screen to assist an improvement of right decisions from the users. To achieve this, we calculated a weight value of MSR classification which indicates Malicious or Safe, and proposed user interface displays information of an application more effectively to the users. Moreover, risk values of each permissions are computed to express the risk compared to the other permissions in an application, and information about detail of permissions are shown in the screen to maximize the users' awareness. The number of samples and accuracy in MSR classification is proportional. However, the number of clean applications dominate malicious ones, an accuracy to detect the malicious application may be decreased compared to the clean. The test of increased attention to install an application with improved design and an accurate test data generation should be executed in further research.

## Acknowledgements

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "IT Consilience Creative Program" (NIPA-2013-H0203-13-1002) supervised by the NIPA (National IT Industry Promotion Agency).

## References

[1] C.-Y. Huang, Y.-T. Tsai and C.-H. Hsu, "Performance Evaluation on Permission-Based Detection for Android Malware", Proceedings of the International Computer Symposium, **(2012)** December 12-14.
[2] I. Rassameeroj and Y. Tanahashi, "A Various Approaches in Analyzing Android Applications with its Permission-Based Security Models", Proceedings of 2011 IEEE International Conference on Electro/Information Technology, Mankato, MN, USA, **(2011)** May 15-17.
[3] L. Kwong Yan and H. Yin, "DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis", Security'12 Proceedings of the 21st USENIX conference on Security symposium, Bellevue, WA, USA, **(2012)** August 8-10.
[4] W. Enck, "Defending Users against Smartphone Apps: Techniques and Future Directions", Proceedings on 7th International Conference, Kolkata, India, **(2011)** December 15-19.
[5] R. Xu, H. Saidi and R. Anderson, "Aurasium: practical policy enforcement for Android applications", Security'12 Proceedings of the 21st USENIX conference on Security symposium, Bellevue, WA, USA, **(2012)** August 8-10.
[6] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev and C. Glezer, "Google Android: A Comprehensive Security Assessment", IEEE Security & Privacy, vol. 8, no. 2, **(2010)**, pp. 35-44.
[7] F. Di Cerbo, A. Girardello, F. Michahelles and S. Voronkova, "Detection of Malicious Applications on Android OS", Proceedings of the 4th international conference on Computational forensics, Tokyo, Japan, **(2010)** November 11-12.
[8] Google. Android; http://www.android.com.
[9] Google. Android permissions; http://developer.android.com/reference/android/Manifest.permission.html.
[10] Rovio. Angry birds; http://www.angrybirds.com/.
[11] Kakao Corporation. KaKaoTalk; http://www.kakao.com/.
[12] B. Skaljac. Monkey Jump 2; http://www.dseffects.com/games/play.php?g=MonkeyJump.
[13] M. Nauman and S. Khan, "Design and Implementation of a Fine-grained Resource Usage Model for the Android Platform", International Arab Journal of Information Technology, vol. 8, no. 4, **(2011)**, pp. 440-448.
[14] A. Porter Felt, E. Chin, S. Hanna, D. Song and D. Wagner, "Android Permissions Demystified", Proceedings of the 18th ACM conference on Computer and Communications Security, Chicago, IL, USA, **(2011)** October 17-21, pp. 627-638.

 

[15] A. Gupta, C. Milanesi, R. Cozza, A. Zimmermann, C. K. Lu, T. Huy Nguyen, H. J. De La Vergne, A. Sato, S. Shen and D. Glenn, "Market Share: Mobile Phones by Region and Country, 3Q12. Gartner", http://www.gartner.com/id=2236115, **(2012)** November 13.
[16] Z. Bu, T. Dirro, P. Greve, Y. Lin, D. Marcus, F. Paget, C. Schmugar, J. Shah, D. Sommer, P. Szor and A. Wosotowsky, "McAfee Threats Report: First Quarter 2012", McAfee Lab **(2012)**; http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2012.pdf.
[17] S. Lee and D. Young Ju, "Assessment of malicious applications using permissions and enhanced user interfaces on Android", Proceedings of the IEEE Intelligence and Security Informatics 2013, Seattle, WA, USA, **(2013)** June 4-7.

## Authors

**Sangho Lee**
2012: School of Integrated Technology at Yonsei University
2007-2012 : BS, Computer Science at Digipen Institute of Technology

**Da Young Ju**
2012: Assistant Professor, Yonsei University, Korea
2002-2011 : PhD, Media Technology, Sogang University, Korea
2007-2008 : MA, Digital Arts, University of the Arts London, UK
2000-2002 : MS, Media Technology, Sogang University, Korea
1995-2000 : BFA, Painting & Design, Hongik University, Korea