# Detection of metamorphic malicious mobile code on android-based smartphones

Sangdon Kim and Hyung-Woo Lee*

School of Computer Engineering,
Hanshin University,
137, Hanshindae-gil, Osan,
Gyeonggi-do 448-791, South Korea
Email: greentea815@naver.com
Email: hwlee@hs.ac.kr
*Corresponding author

Jae Deok Lim and Jeong Nyeo Kim

Cyber Security-Convergence Research Laboratory,
ETRI, 34129, South Korea
Email: jdscol92@etri.re.kr
Email: jnkim@etri.re.kr

**Abstract:** By repackaging a malicious code into reverse compiled legitimate mobile code, malware authors can bypass detection step on existing mobile vaccine software using inserting AES-encrypted root exploits to loading some payload from a malicious remote server dynamically. In this case, malicious codes are constantly changing to evade detection steps by continuing its evolution by operating a metamorphic code by adding new propagation vectors, functionality, and stealth techniques to hide its presence and evade the detection of antivirus software. Those metamorphic features are aimed at changing the form of each instance of the malware by using encryption or appended/pre-pended dummy code into internal code of mobile apps. Therefore, we propose a new system to determine and detect metamorphic malicious mobile code by extracting dynamic features activated from Android platform using extended dynamic analysis technique.

**Keywords:** metamorphic malicious mobile apps; dynamic analysis; android; smartphone and detection of malicious apps.

**Reference** to this paper should be made as follows: Kim, S., Lee, H-W., Lim, J.D. and Kim, J.N. (2017) 'Detection of metamorphic malicious mobile code on android-based smartphones', *Int. J. Advanced Media and Communication*, Vol. 7, No. 1, pp.56–75.

**Biographical notes:** Sangdon Kim received his MS degree in Computer Engineering from Hanshin University, Korea in 2012. His research interests include mobile security and system security.

Hyung-Woo Lee received his BS, MS and PhD in Computer Science from Korea University, Seoul, Korea, in 1994, 1996 and 1999, respectively. Currently, he is a Professor at the Division of Computer Engineering, Hanshin University, Gyyeong-gi Province, Korea. His research interest is in the areas of network Security, cryptography and computer forensics.

Jae Deok Lim received his MS degree in Electronic Engineering from Kyungbook National University, Korea, in 2001 and the PhD in Computer Engineering from Chungnam National University, Korea, in 2013. He is currently a Member of Engineering Staff at Information Security Research Division in ETRI (Electronics and Telecommunications Research Institute). His research interests include IoT Security, mobile security, access control, secure operating system and system security.

Jeong Nyeo Kim received her MS degree and PhD in Computer Engineering from Chungnam National University, Rep. of Korea, in 2000 and 2004, respectively. She studied at computer science from the University of California, Irvine, USA in 2005. Since 1988, she has been a Principal Member of engineering staff at Information Security Research Division in ETRI (Electronics and Telecommunications Research Institute. Her research interests include IoT security, mobile security, secure operating system, network security and system security.

This paper is a revised and expanded version of a paper entitled 'New approach to determine metamorphic malicious mobile code on android-based smartphones' presented at *ICONI2014*, Taipei, Taiwan, 14–16 December, 2014.

## 1 Introduction

Because the number of users of Android-based commercial smartwork devices is increasingly rapidly, new types of applications (apps) are being developed and distributed in the Android market. However, in addition to the distribution of normal apps, the distribution of malicious apps with malicious purpose is also increasing rapidly. Smartphones are becoming more popular everyday and they are considered as an essential part of our life. Furthermore, they are starting to replace traditional desktop computers because of their superior mobility and nearly omnipresent wireless internet access. Android is getting a lot of attention, and its popularity is growing exponentially from day-to-day. But Android malware is evolving quickly. Malware writers use all sorts of tricks to avoid detection, and one of those is called metamorphism that allows code to change without changing what the code actually does (Leder et al., 2009).

By repackaging a malicious code into reverse compiled legitimate mobile code, malware authors can bypass detection step on existing mobile vaccine software using inserting AES-encrypted root exploits to loading some payload from a malicious remote server dynamically. In this case, those kinds of malicious codes are constantly changing to evade detection steps by continuing its evolution by operating a metamorphic code by adding new propagation vectors, functionality and stealth techniques to hide its presence and evade the detection of antivirus software. Android devices will get infected with malicious codes through changes of access permission. Mobile malicious apps based on Android which leaks the personal and financial information by causing malfunction and consuming the batteries of devices have consistently been increasing (http://securelist.com/analysis/kaspersky-security-bulletin/58335/mobile-malware-evolution-2013/, Uscilowski, 2003).

Therefore, techniques (http://www.eweek.com/security/slideshows/security-stats-show-mobile-malware-xss-as-top-concerns.html, http://nakedsecurity.sophos.com/2012/07/31/server-side-polymorphism-malware/) monitoring malicious app events have been presented to detect the intrusion towards mobile devices in a bid to reduce damages through spread of malicious app like this, but mechanism should be developed to discriminate malicious apps from normal apps of commercial mobile devices. Therefore, it is necessary to analyse the newly issued attack mechanism such as metamorphic malicious mobile apps and analyses the characteristics of metamorphic type of malicious apps with activation pattern activated from Android platform. We propose a method for determining intelligent metamorphic malicious apps, which has recently become an issue, by applying a dynamic analysis technique for Android platform-based malicious apps. In detail, we proposed a hidden evasive type of metamorphic malicious mobile code detection mechanism by aggregating dynamic system call events and analysing its feature on Android smartphone devices.

## 2 Malicious mobile apps

The Android platform's openness is part of what makes it attractive to users, because it offers more control over their devices and the apps they install. Furthermore, everyone gets the chance to post an application for download in the Android open market without any previous application reviews from Google who owns it. But the freedom is also the downside of the platform: cyber criminals exploit it by posting malware-spreading apps in devious attempts to steal personal information. In this reason, the users' mobile security is clearly compromised. Therefore, users who download it unknowingly face a mobile privacy threat. Android's increase in popularity and its openness have triggered a great rise in malware-spreading apps via the Android open market. The most common Android malicious apps contain spyware and (SMS) Trojans as a hidden undetectable code. Those apps collect and send GPS coordinates, contact lists, email addresses to third parties. And they record phone conversations and send them to attackers. Furthermore, malicious apps download other malware onto infected phones using advanced deformation and transformation tricks based on an existing exploit such as a metamorphism (Felt et al., 2011).

The mobile malware sector is growing rapidly both technologically and structurally. There are various types of actors involved in the mobile malware industry. More than 150,000 malicious mobile apps were detected in 2013. In detail, cyber criminals to distribute mobile malware use 4 million packages. Overall in 2012–2013 approximately 10,000,000 unique malicious installation packages are detected. Different installation packages can install programs with the same functionality that differs only in terms of the malicious app interface and, for instance, the content of the text messages it spreads (http://securelist.com/analysis/kaspersky-security-bulletin/58335/mobile-malware-evolution-2013/). Most malware are designed to steal user's money and lots of backdoors. Figure 1 shows the trend in the percentage of malware in the largest app stores. Android remains a prime target for malicious attacks. 98.05% of all malware detected in 2013 targeted this platform, confirming both the popularity of this mobile OS and the vulnerability of its architecture.

Recently, mobile malware writers use all sorts of tricks to avoid detection, and one of powerful tricks is to use polymorphism inside of Android mobile apps. It is a cool trick that allows code to change without changing what the code actually does. A new form of metamorphism, as an advanced polymorphism, has been used to evade detection on PC based OS like Windows systems, furthermore, an advanced mobile malware has also been discovered targeting the Android platform (Uscilowski, 2013). Smartphone is seemingly under constant attack from all corners of the globe at all times. Reports out in recent weeks provide fresh fuel for the analysis of mobile security and what is under attack. Among the major trends is the fact that mobile malware continues to rise. In Figure 2, data from F-secure shows strong growth for malware, spyware and adware on Android (http://www.eweek.com/security/slideshows/security-stats-show-mobile-malware-xss-as-top-concerns.html).

**Figure 1** The distribution of mobile malware detected in 2013 by platform (see online version for colours)

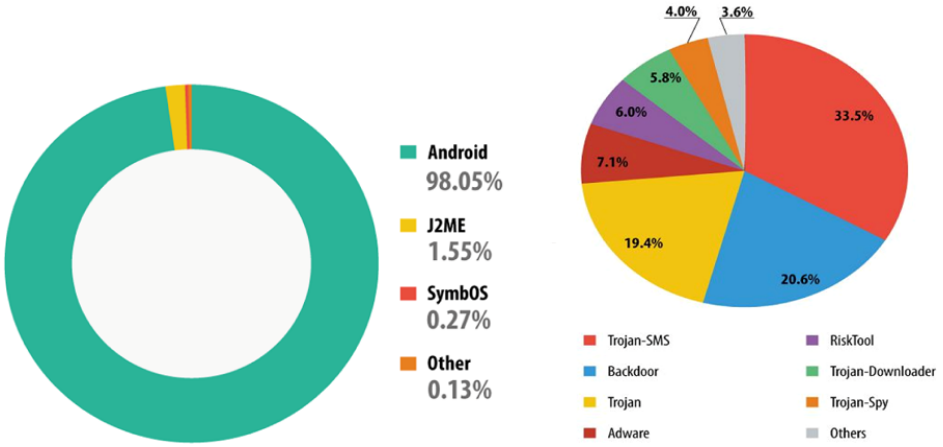
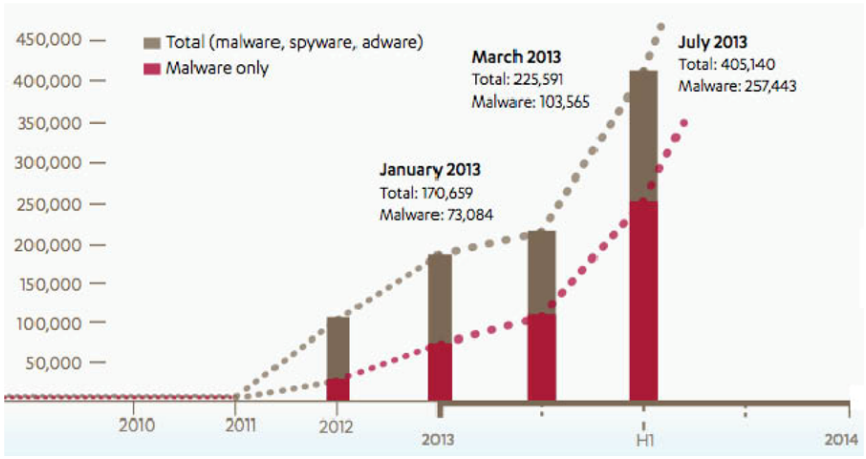



**Figure 2** Android mobile malware detected in 2013 (see online version for colours)



## 3 Metamorphic malicious mobile apps on android

### *3.1 Polymorphic malicious code for metamorphic*

Metamorphic and polymorphic malware are two categories of malicious software programs (malware) that have the ability to change their code as they propagate. Metamorphic malware is rewritten with each iterations so that each succeeding version of the code is different from the preceding one. Therefore, these metamorphic and polymorphic code changes make it difficult for existing signature-based antivirus software programs to recognise that different iterations are the same malicious program. Polymorphic malware (http://nakedsecurity.sophos.com/2012/07/31/server-side-polymorphism-malware/) also makes changes to code to avoid detection. It has two parts, but one part remains the same with each iteration, which makes the malware a little easier to identify. In another example, a new key might be randomly generated with each copy to change the appearance of the encrypted virus body – but the virus decryption routine would remain constant. In this scenario, it is the static part of the code that makes it possible for an antivirus program to identify the presence of malware. Therefore, it is impossible for current signature-based known-malware detection technology to reliably detect all instances of a piece of malware that employs server-side polymorphism (http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/detecting-polymorphic-malware).

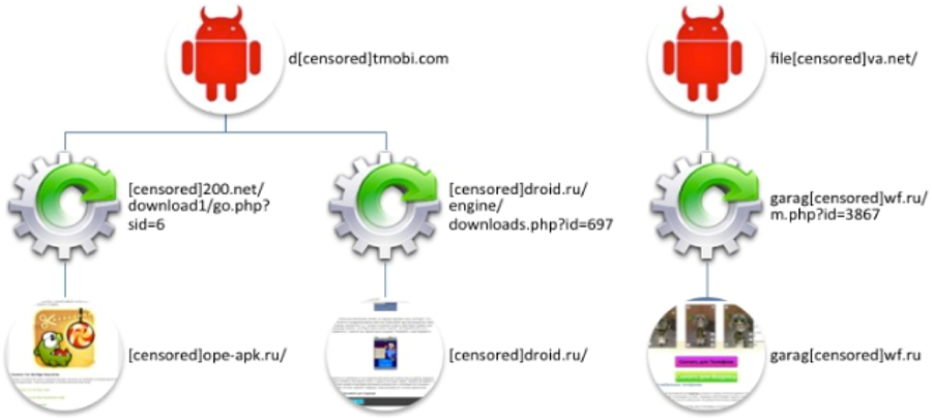
### *3.2 Metamorphic code on android*

In spite of the permanent changes to code, each iteration of metamorphic malware functions the same way. The longer the malware stays in a mobile device, the more iteration produces and the more sophisticated the iterations are quarantine and disinfect, which makes it increasingly hard for antivirus applications to detect. Metamorphic malware is considered to be more difficult to write than polymorphic malware. The author may use multiple transformation techniques, including register renaming, code permutation, code expansion, code shrinking and garbage code insertion. Consequently, advanced techniques such as generic decryption scanning, negative heuristic analysis, emulation and access to virtualisation technologies are required for detection.

Advanced metamorphic malware is malicious software that is capable of changing its code and signature patterns with each iteration. At its very simplest, malware contains an encrypted virus that has a decryption routine (VDR) and an encrypted virus body (EVB). When the infected application executes, the VDR decrypts the EVB and the virus carries out its intended function. In the propagation phase, the virus gets reencrypted and attached to another host application. Each copy generates a new key, but the virus decryption routine remains the same and this is how antivirus software applications can identify malware programs (Figure 3).

Polymorphic malware adds an additional component to the encrypted code as a mutation engine (ME) that changes the virus decryption routine with each iteration by using obfuscation techniques such as inserting junk code, reordering instructions and using mathematical contra-positives. This type of malware can still be recognised by antivirus software fairly easily, however, because the decrypted virus body remains the same.

**Figure 3** Server-side polymorphic apps: daily creation of new fake websites (see online version for colours)




Metamorphic malware takes virus mutation to the next level. It uses the polymorphic malware's mutation engine to change both the virus decryption routine and the EVB as in Figure 4. The mutation engine disassembles the code and represents it with a meta-language that characterises the code's function but disregards how the code achieves this function. The end result is the new code that bears no resemblance to its original syntax, but is functionally the same. Metamorphic malware is more difficult for antivirus software to recognise, but not impossible. The weakness of metamorphic software is that the mutation engine needs to analyse the code in order to disassemble it and if the mutation engine can analyse the code, so can vendors who make antivirus software (http://blogs.mcafee.com/mcafee-labs/fakeinstaller-leads-the-attack-on-android-phones).

A user unknowingly goes through the process of registering in a C&C server operated by an attacker; afterwards, the C&C server generates a different type of malicious code every time and stores it in the app download server. Then, the user downloads a different type of malicious code from the pertinent server every time and installs it in the user's device. Because the mobile apps downloaded by each user who undergoes this process are written with slightly different code, they have the characteristic of not being detected though conventional mobile detection methods.

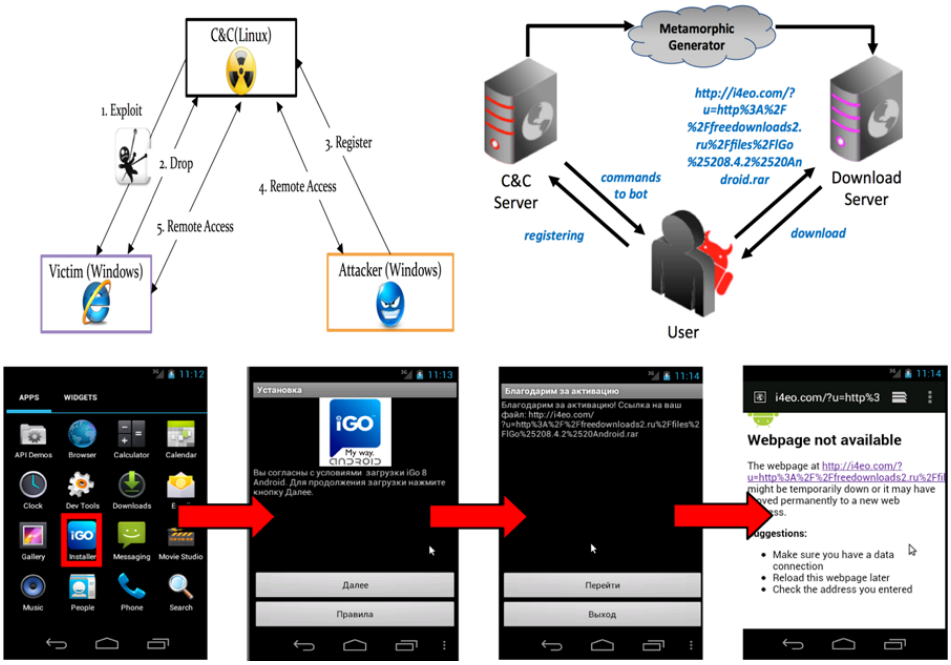
### *3.3 Characteristics of metamorphic malicious apps*

Most metamorphic codes discovered at present are modified malicious codes of the fake installer (FakeInst) and OpFake types. With most of these, different types of code values are downloaded every time through linkage with a remote server. When the operating method for the FakeInst type malicious app is examined, we can confirm that an SMS message is sent to an external server designated by the attacker unperceived by the user (Ham et al., 2014).

Similar to the FakeInst type, the OpFake type of polymorphic malicious apps can be confirmed to send SMS messages to an external server unperceived by the user; when the internal code is analysed, it could be confirmed that an obfuscation function is applied

with respect to the information contained in XML format inside the Android mobile apps. In detail, the internal code part concerned with sending and receiving SMS is encrypted confirmatively.

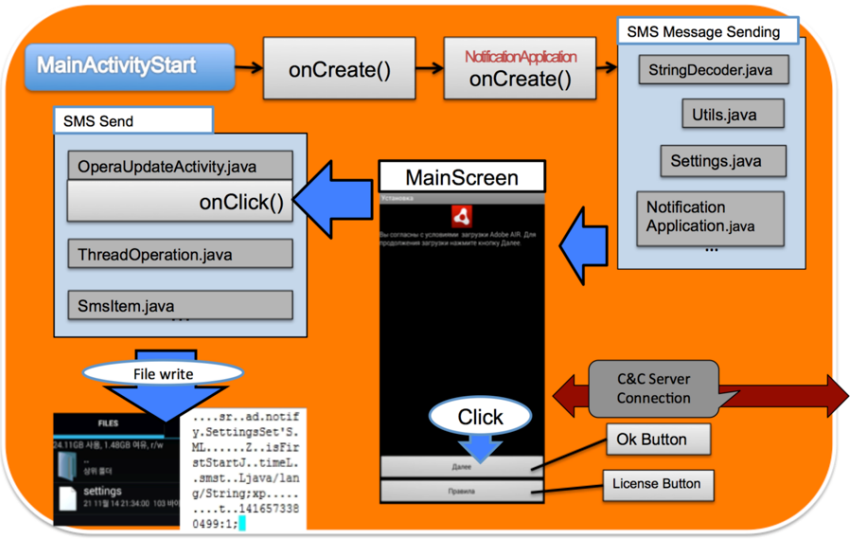
**Figure 4** Activation mechanism of metamorphic malicious code and its apps (see online version for colours)




When metamorphic code is installed in each user device, it operates such that it has mutually different signature values. Therefore, in the devices of different users, it is installed with values different from the known signatures of malicious codes, and consequently, it has the characteristic of avoiding the detection process of mobile detection methods. By applying an encryption for the string information value declared in the internal codes, it is hard to detect an obfuscation function from suspicious metamorphic mobile apps. Furthermore, they are devised to perform different types of execution processes every time by applying an advanced mechanism, such as changing the execution sequence for internal codes of the mobile app through the use of random numbers selected arbitrarily. Therefore, when this function is applied, they show a characteristic of not being detected through mobile detection methods.

Using the random number table defined autonomously, the text strings were recorded in encrypted form inside the source codes. When this function is used, a characteristic of evading the packet-based monitoring process is included because encrypted or transformed strings are transmitted with respect to the information sent and received through the network (Figure 5).

**Figure 5** Internal functions of metamorphic malicious code (see online version for colours)




## 3.4 *Static analysis of metamorphic malicious apps*

To analyse metamorphic malicious apps, a static analysis could be performed first using the existing 'Androguard' tool (https://code.google.com/p/androguard/). To install Androguard, an experimental environment was established using VMware on the Ubuntu Version 11.10 operating system. Figure 6 shows the final outcome of the Androguard static analysis environment.

An Androguard file consists of several Python files. The internal structure of the Androguard can be largely divided into three directories: core, decompiler and patch modules; in the decompiler directory, there are several Python files, such as androapkinfo.py, androarsc.py and androauto.py. The core directory also provides functions such as analysis, binaries, bytecodes, data and a debugger. There are Python scripts such as androconf.py, androgen.py and bytecode.py. The analysis consists of Python files that provide such static analysis functions. There are 13 tools in Androguard. Specially, Apkviewer.py shows information about APK files and Androguard.py includes a total of 13 Python codes and modules; these play an important part in making static analysis. The order of Androguard is as follows: first, the subject APK file of analysis is called. Then, the binary XML is converted to the original XML file (one that a person can read), after which the file information of XML is extracted, followed by a decompilation of the Dex file. Next, the class and method included in the decompiling process is extracted. Finally, the data are generated or shown in graph format.

## 3.5 *Dynamic analysis of metamorphic malicious apps*

DroidBox (https://code.google.com/p/droidbox/) is developed to offer dynamic analysis of Android applications. The following information is shown in the results, generated when analysis is ended:

- hashes for the analysed package
- incoming/outgoing network data
- file read and write operations
- started services and loaded classes through DexClassLoader
- information leaks via the network, file, SMS, etc.

Many forms of tools can carry out a dynamic analysis on Android apps. Of these, this study used DroidBox to analyse the characteristics of polymorphic malicious apps as Figure 7. As DroidBox provides information on the internal actions of the Android app on the basis of live-feeding execution processes in graph form, it can be widely used to understand the internal actions of metamorphic malicious apps.

**Figure 6** Establishing an androguard-based static analysis environment (see online version for colours)


**Figure 7** Dynamic analysis on droidBox-based polymorphic malicious apps (see online version for colours)

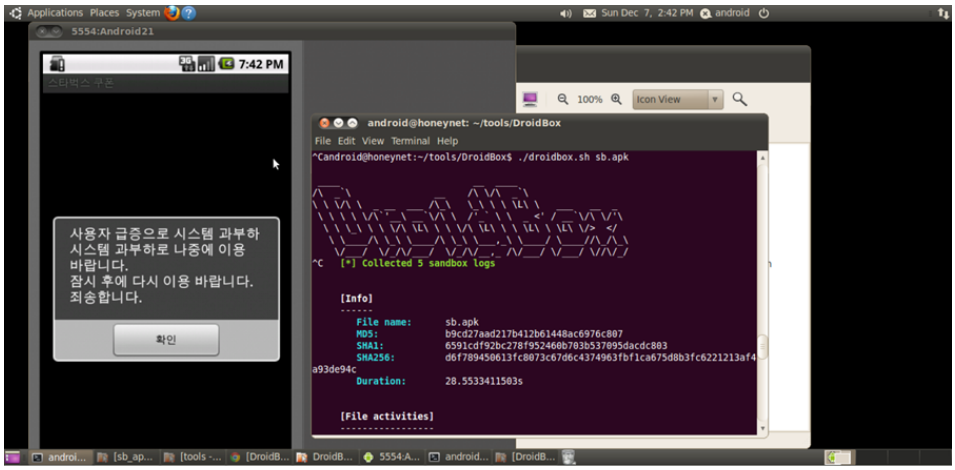
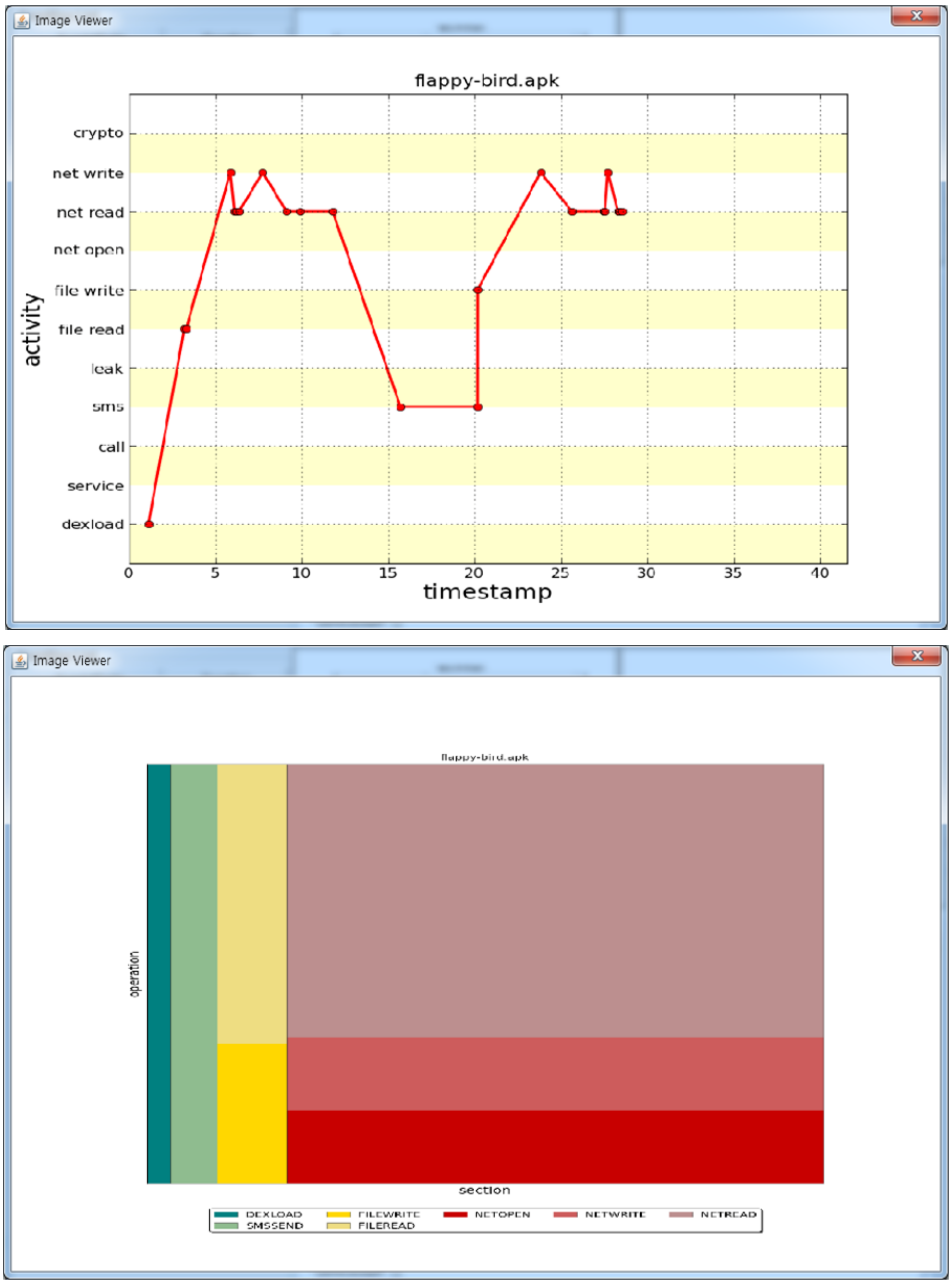


**Figure 7** Dynamic analysis on droidBox-based polymorphic malicious apps (see online version for colours) (continued)




Using DroidBox, an analysis could be performed on the internal event and network connecting process that takes place while a polymorphic malicious app is running. However, because using dynamic analysis functions of DroidBox does not provide a perfectly effective function for detecting polymorphic malicious apps, the functions were improved in this study through the analysis of the internal module structure of the

existing DroidBox, as to allow analysis and differentiation of polymorphic malicious apps. More specifically, the points for improvement were deduced by carrying out an analysis on the functions provided by DroidBox, and on the basis of this analysis, it was possible to develop an improved integrated analysis tool for Android malicious apps.

The core process of DroidBox is live-feed analysis on the characteristics of events that occur within the Android terminal, such as DexClassLoader, ServiceStart and RecNet, based on the LogCat information that is generated when the app starts running, and on the process of differentiating these characteristics. Therefore, because it is possible to understand the activities of the app on the basis of the live-feed LogCat information and because they are represented in graph form, it can be used in the differentiation process of malicious apps.

## 4 Proposed metamorphic malicious code (MMC) detection system

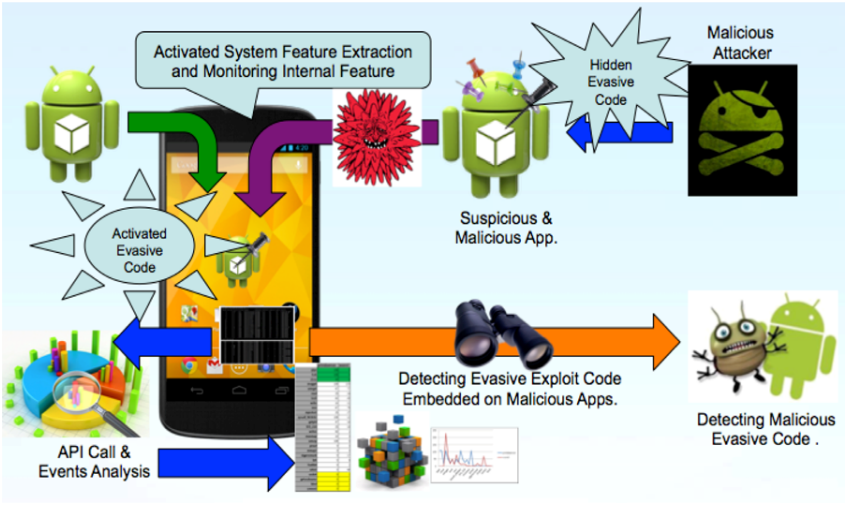
### *4.1 Metamorphic malicious code (MMC) detection model*

To detect a metamorphic type of malicious mobile code, we constructed a dynamic system call event monitoring and analysis system. We implemented an updated dynamic system event aggregation (uDEA) application. The uDEA app implemented in this study enables a user to verify and retrieve services and processes running on inside of kernel in Android-based mobile device. Used module invokes event-monitoring daemon at kernel and the application is running in background. Database (DB) server is implemented to collect and store the event data that come from multiple devices. Those events information collected from the mobile device are used to detect any suspected event due to running a malicious MMC code. In first, we aggregated system call event from 120 kinds of polymorphic-type of malicious apps. On the basis of this dataset, we can extract some characteristics of metamorphic type of malicious apps. In detail, we can find the different sequence pattern of metamorphic malicious mobile apps. Therefore, we can classify malicious metamorphic mobile apps from normal ones (Pandi et al., 2011).

The purpose of the proposed model is to effectively detect evasive metamorphic malicious apps on Android mobile devices. The proposed method is different from the existing scheme in the sense that we can monitor and detect the hidden evasive type of MMC on multiple devices dynamically. The existing DroidLogger mechanism (Dai et al., 2012) proposed a logging-based analysis and detection mechanism. But, we proposed a malicious code detection mechanism operated as a MMC by analysing dynamic system call events as shown in Figure 8. The proposed scheme enables to aggregate real-time system call events activated from each user's Android smartphone devices so that we can determine malicious apps from normal one.

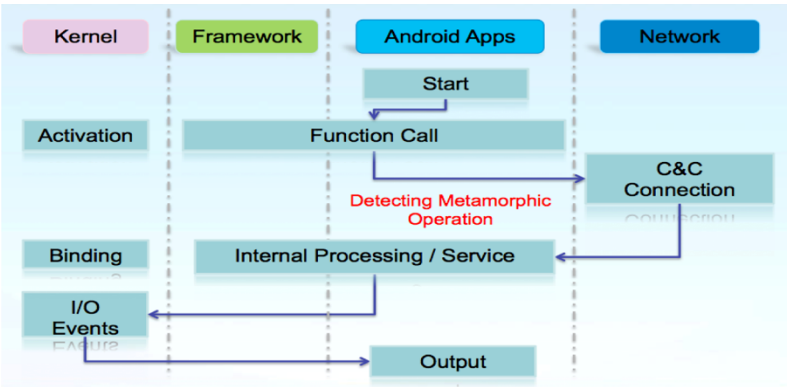
We can define a sequence of length $m$ over the element $T$ as a formal definition such as $S = s_1, \ldots, s_m$ where each $s_i$ is a member of $T$ and is called an element of $S$ (https://code.google.com/p/droidbox/, Dai et al., 2012). Sequences from mobile apps play core roles to determine whether it is malicious or not. Therefore, measuring sequence similarity is an important part of application such as identifying the characteristics patterns of system call event from suspicious unknown mobile apps.

**Figure 8** Proposed metamorphic malicious code detection model (see online version for colours)




As shown in Figure 9, we can aggregate system call activated from MMC. And then, we can estimate its logical distance between malicious and normal ones. For estimating its difference, we should perform the pre-processing step such as a sequence adjustment to identify regions of similarity that may be a consequence of functional relationships between the sequences for estimating logical distance on aggregated system call events. In case of malicious mobile apps, we can see the difference pattern of metamorphic malicious mobile apps, which have a random length of empty gaps by extracting related events. In detail, we can estimate sequence distance primitives after performing alignment process on aggregated system call event set.

**Figure 9** Metamorphic events detection step (see online version for colours)




## *4.2 MMC detection system*

By analysing the functions of the pre-existing DroidBox, an improved integrated dynamic analysis tool that provides differentiation functions of polymorphic malicious apps was developed. As Figure 10 shows, by understanding the problems of the internal module

structure of the pre-existing DroidBox, an improved linkage diagram of the module could be made. In addition, as Figure 11 shows, the function was expanded to include one that would allow DB management of metamorphic malicious apps. Through this, the polymorphic malicious app DB group was constructed, and a basic environment was created that enhanced the accuracy of differentiating random apps.

**Figure 10** Function interface improvement for integrated dynamic analysis (see online version for colours)

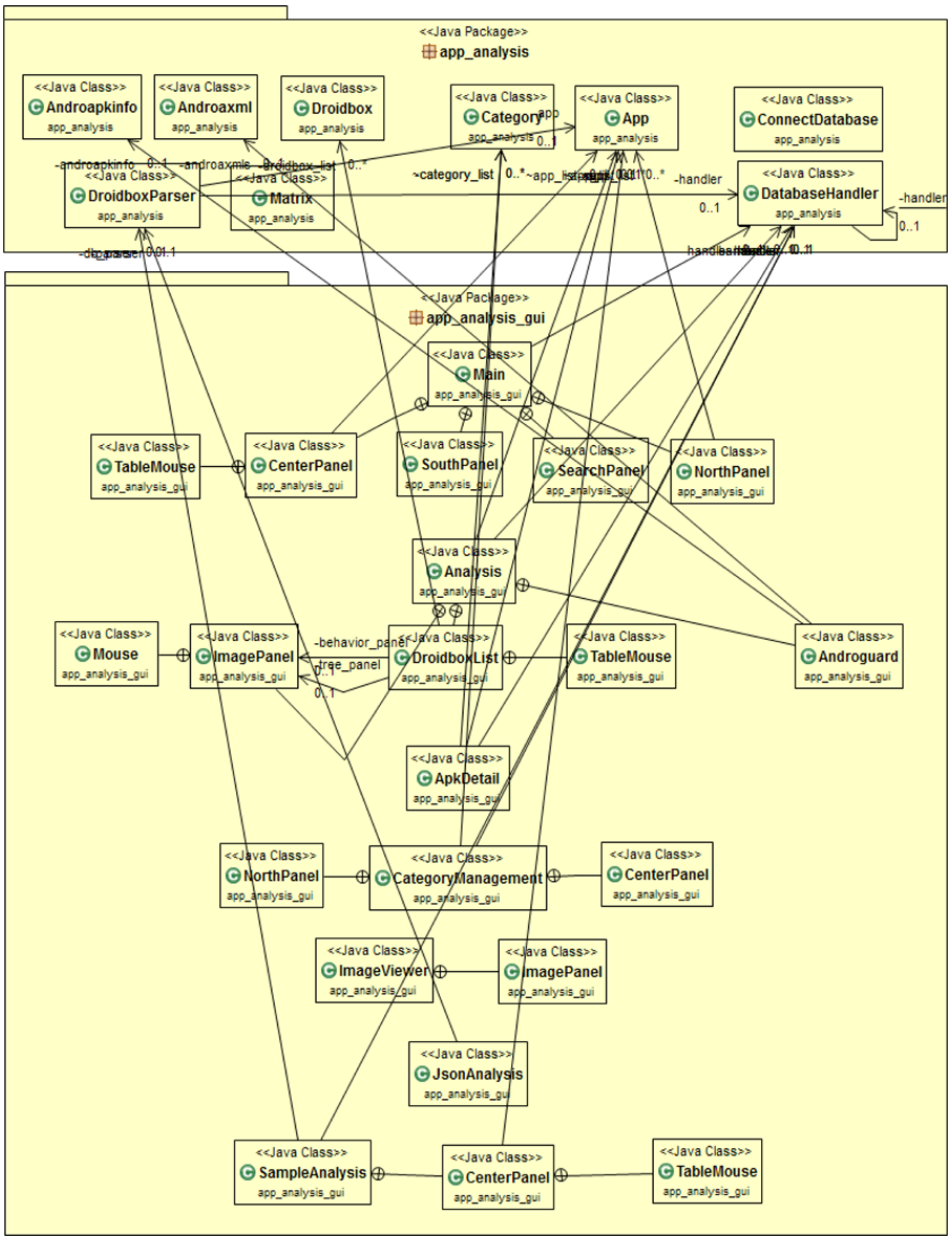
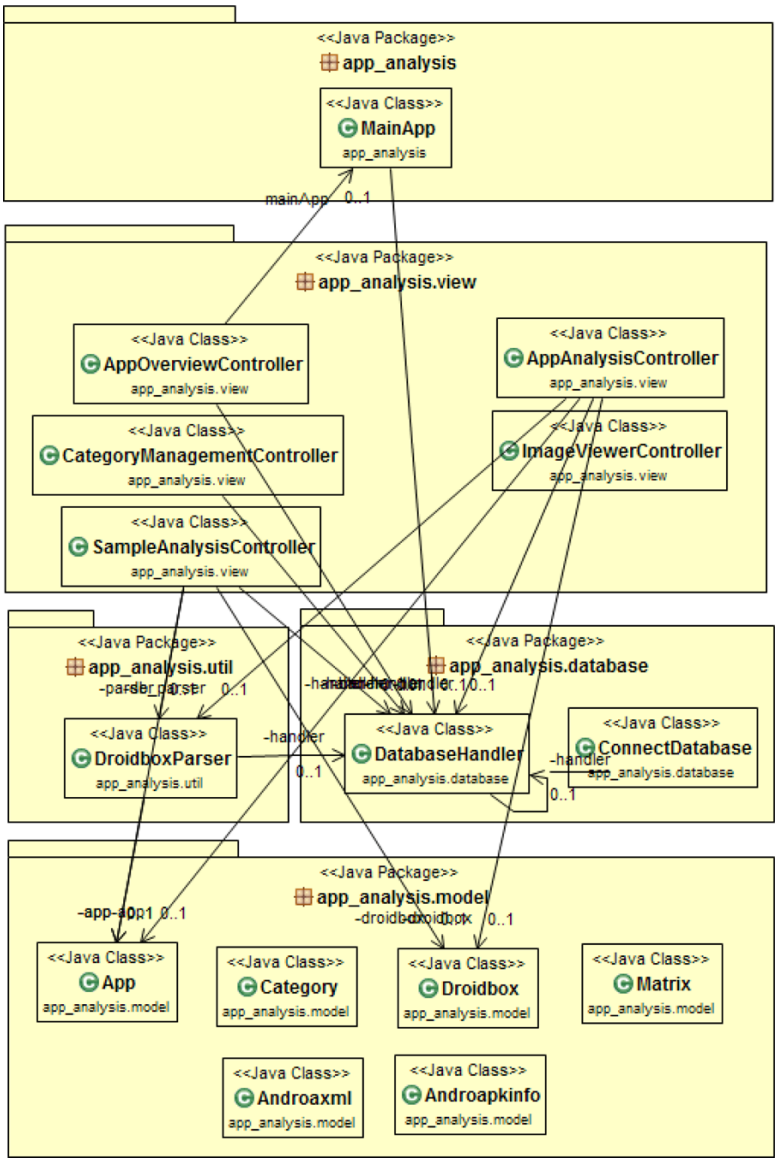


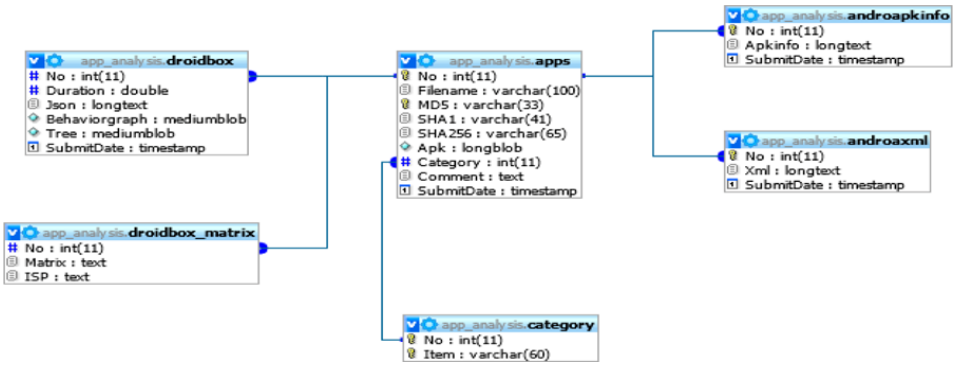
**Figure 10** Function interface improvement for integrated dynamic analysis (see online version for colours) (continued)




A DB schema that would provide not only the information provided by Droidbox as well as dynamic analysis functions, but also the differentiation functions of the app, was designed and developed to establish the DB of polymorphic malicious apps.

The results of dynamic analysis tool development on polymorphic malicious apps are as follows. The functions of the pre-existing Androguard and DroidBox modules were utilised and integrated, and they were altered/improved in order to provide analysis and differentiation functions on metamorphic malicious apps (Figure 12).

**Figure 11** DB schema for dynamic analysis of mobile apps (see online version for colours)




The proposed dynamic analysis-based detection system can be applied to classify metamorphic malicious apps, which have recently been spreading rapidly. In the case of applying, the dynamic event-oriented malicious app determination system proposed in this study, first, a DB building function is provided for intelligent and new types of those variant malicious apps that have recently been spreading rapidly, and based on this, even if a new variant of malicious app is developed, more active responses are possible using the intelligent malicious app DB and determination mechanism constructed through this study. Therefore, it is possible to apply the proposed dynamic analysis-based malicious app determination technique in the smartphone device environment.

## 5 Experimental results

### *5.1 Dynamic analysis of metamorphic malicious code*

By using the integrated dynamic analysis tool, it was possible to carry out an analysis and differentiation process of polymorphic malicious apps collected via the internet. It can not only modify the core functions of the pre-existing DroidBox module and primarily establish a group of data on metamorphic malicious apps through establishing DB, but also provide the added advantage of enhancing accuracy of the differentiation process of random apps, using the established DB information.

To analyse the performance of the process of analysis and differentiation of polymorphic malicious apps, an integrated dynamic analysis was carried out on the malicious app Com.adobe.flashplayer.apk as Figure 13 shows.

By carrying out a static analysis on the subject app, the permission information included in the app could be automatically extracted. Using proposed system, it was also possible to find an illegal access module included within the terminal internal file by analysing the internal code of com.adobe.flashplayer.apk, and it included a function of sending information on the malfunctioning of the terminal and the state of the mobile phone. In addition, through static analysis, it was possible to obtain the internal codes of com.adobe.flashplayer.apk; thus, modules or functions with suspected

abnormality were automatically extracted. The results of static and dynamic analysis could be presented in an integrated form along with the permission information included within the app.

**Figure 12** Metamorphic malicious apps dynamic analysis tool (see online version for colours)

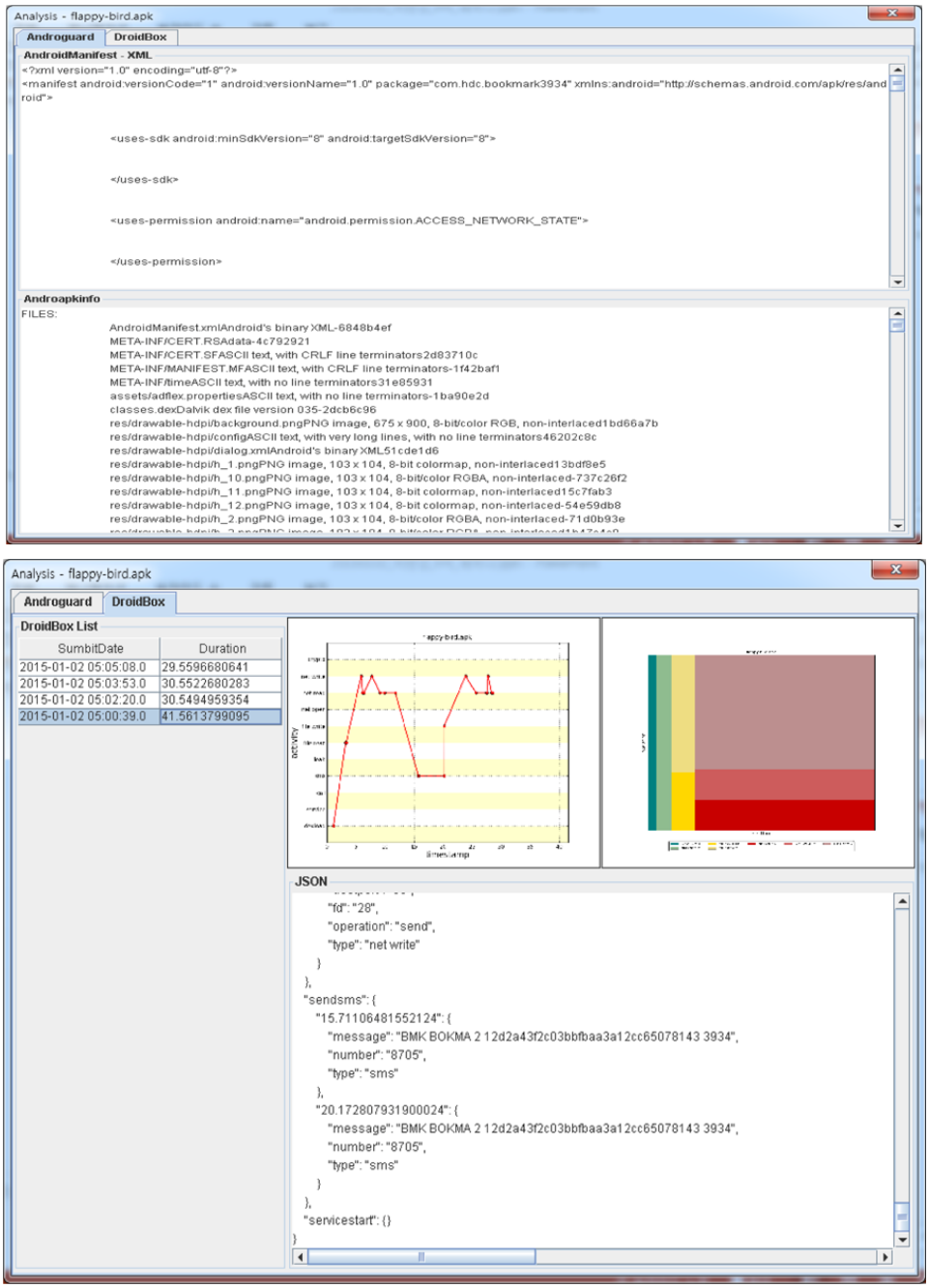
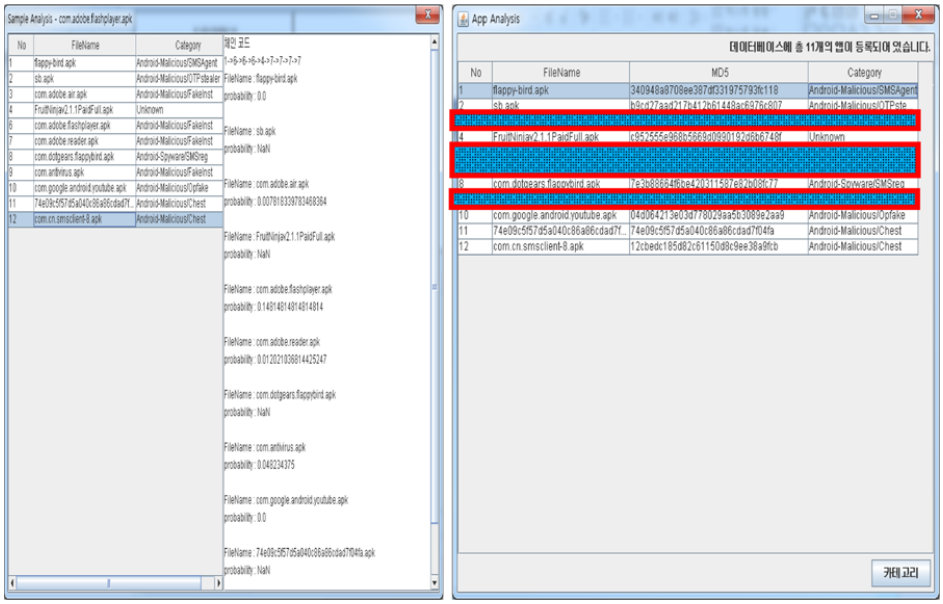


**Figure 13** Dynamic analysis target apps (com.adobe.flashplayer.apk) (see online version for colours)


## 5.2 *Detection of metamorphic malicious code*

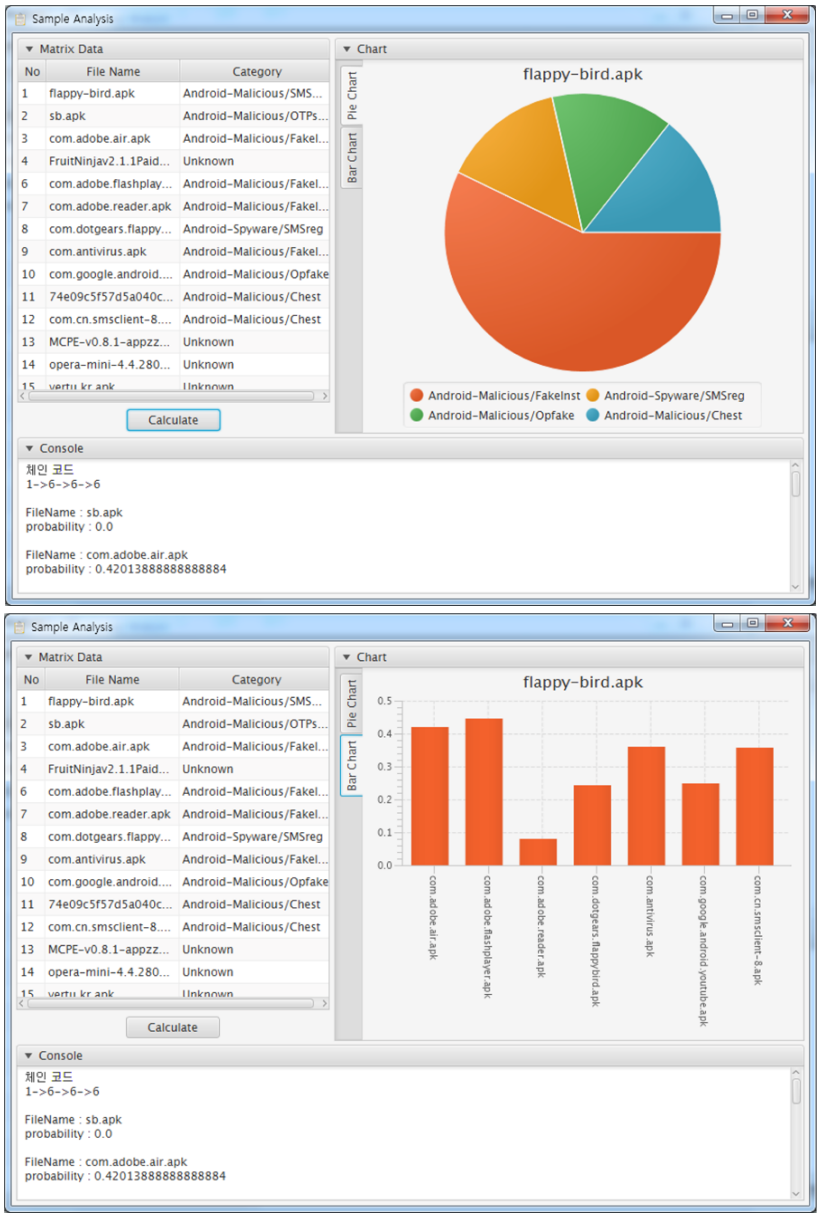
In this study, experimental data were acquired by establishing a polymorphic malicious app DB, and the characteristics of internal events that occur when a polymorphic malicious app is running were acquired through altering/improving static and dynamic analysis modules. It was also possible to see that out of all the subject apps, four apps were highly similar to the polymorphic malicious app group in FakeInst form (http://blogs.mcafee.com/mcafee-labs/fakeinstaller-leads-the-attack-on-android-phones). This method ultimately identified it as a polymorphic malicious app. In addition, on this basis of information from the behaviour graphs, an underlying execution pattern in all polymorphic malicious apps was found (Figure 14).

**Figure 14** Detection results of metamorphic malicious apps (see online version for colours)



To make a more precise distinction on whether the events found in suspicious apps are closer to MMC, event distance and its sequence difference can be estimated by monitoring internal processing calls. Therefore, we suggested a metamorphic malicious application distinction method through by calculating distance value using extracted common identity code to estimate the sequence similarity rate that can distinguish metamorphic malicious application. Simple experimental results are shown in Figure 15 after using dynamic activity similarity measures based on aggregated real-time system calls.

**Figure 15** Metamorphic malicious apps event pattern (see online version for colours)



## 6 Conclusions

As the number of Android platform-based mobile device users increases rapidly, the number of malicious apps that leak users' personal and financial information to external servers is also increasing rapidly. These malicious apps are installed and operated on user devices by camouflaging as normal apps, providing the same shape and function of existing normal apps. Furthermore, the latest malicious apps have various evasive metamorphic functions to avoid detection through conventional mobile detection methods. Therefore, to improve detection performance with respect to these intelligent malicious apps, it is necessary to perform static and dynamic analyses for normal and malicious apps, and based on this, provide a method for effective detection of suspicious malicious apps.

Metamorphic malware is rewritten with each iterations so that each succeeding version of the code is different from the preceding one. Therefore, metamorphic code changes make it difficult for existing signature-based antivirus software programs to recognise that different iterations are the same malicious program. We propose a new approach to determine metamorphic malicious mobile code by monitoring and extracting dynamic system call features activated from Android kernel. On the basis of those characteristics, we can classify and detect metamorphic malicious mobile code efficiently using optimised system call sequence similarity measure aggregated from Android kernel level. To detect metamorphic malicious mobile apps, we perform two kinds of analysis processes. If anonymous test app is decided to a malicious app, then we perform the sequence analysis step to measure the sequence distance of system call event pattern with that of known MMC mobile apps.

## Acknowledgements

This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2014R1A1A2057430).

## References

Dai, S., Wek, T. and Zou, W. (2012) 'DroidLogger: reveal suspicious behavior of Android applications via instrumentation', *International Conference on Computing and Convergence Technology (ICCCT)*, pp.550–555.

Felt, A.P., Finifter, M., Chin, E., Hanna, S. and Wagner, D. (2011) 'A survey of mobile malware in the wild', *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, *SPSM '11* , ACM, New York, NY, USA, pp.3–14.

Ham, Y.J., Moon, D., Lee, H-W., Lim, J. and Kim, J.N. (2014) 'Android mobile application system call event pattern analysis for determination of malicious attack', *International Journal of Security and Its Applications (IJSIA)*, Vol. 8, No. 1, pp.231–246.

Leder, F., Steinbock, B. and Martini, P. (2009) 'Classification and detection of metamorphic malware using value set analysis', *2009 4th International Conference on Malicious and Unwanted Software (IEEE MALWARE) 2009*, pp.39–46.

Pandi, M.H., Kashefi, O. and Minaei, B. (2011) 'A novel similarity measure for sequence data', *Journal of Information Processing Systems*, Vol. 7, No. 3, September, pp.413–424.

Uscilowski, B. (2013) *Mabile Adware and Malware Analysis*, Security Response, Synamtec.

Zhou, W., Zhou, Y., Jiang, X. and Ning, P. (2012) 'DroidMOSS: detecting repackaged smartphone applications in third-party android marketplaces', *Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy*, ACM, New York, pp.317–326.

## Websites

http://blogs.mcafee.com/mcafee-labs/fakeinstaller-leads-the-attack-on-android-phones

http://blogs.mcafee.com/mcafee-labs/fakeinstaller-leads-the-attack-on-android-phones

http://nakedsecurity.sophos.com/2012/07/31/server-side-polymorphism-malware/

http://securelist.com/analysis/kaspersky-security-bulletin/58335/mobile-malware-evolution-2013/

http://www.eweek.com/security/slideshows/security-stats-show-mobile-malware-xss-as-top-concerns.html

http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/detecting-polymorphic-malware

https://code.google.com/p/androguard/

https://code.google.com/p/droidbox/