International Journal of u- and e- Service, Science and Technology
Vol.8, No. 11 (2015), pp.295-302
http://dx.doi.org/10.14257/ijunesst.2015.8.11.29

Dynamic Analysis System for Detecting Remote Server-Side Polymorphic Malicious Mobile Apps on Android based Smartphone

Han Seong Lee and Hyung-Woo Lee

Division of Computer Engineering, Hanshin University, Yangsan-dong, Osan, Gyyeonggi, 447-791, Rep. of Korea
jkhanseong@naver.com, hwlee@hs.ac.kr

Abstract

As Android malware is evolving quickly, malware creators are starting to develop new kinds of threats such as remote server-side polymorphic malicious code for Android platform that are being actively generated and distributed via the official Android Markets. Remote server-side polymorphic mobile apps can't be detected correctly as those apps contain spyware and trojans as a hidden undetectable code. Furthermore, these types of malicious apps download other malware onto infected phones using advanced deformation and transformation tricks based on an existing exploit. Therefore, we designed and implemented dynamic analysis system to detect evasive and transformative remote SSP malicious mobile apps efficiently. In particular, we proposed web based analysis and management system to validate and confirm suspicious remote server-side polymorphic malicious apps efficiently.

***Keywords**: Dynamic Analysis, Remote Polymorphic Malicious Apps, Android, Smartphone*

1. Introduction

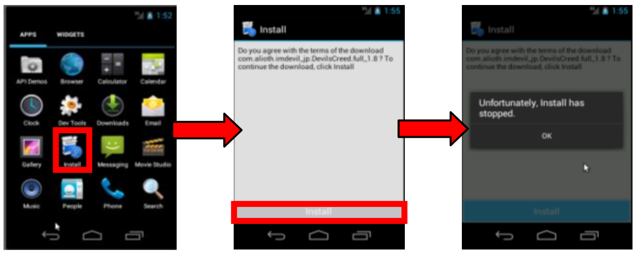
Because the number of users of Android-based commercial smart work devices is increasingly rapidly, new types of applications (apps) are being developed and distributed in the Android Market. However, in addition to the distribution of normal apps, the distribution of malicious apps with malicious purpose is also increasing rapidly. 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, e-mail 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 remote server-side polymorphism (SSP) [1, 2].

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's a cool trick that allows code to change without changing what the code actually does. A new form of polymorphism, as an advanced remote server-side polymorphism, has been used to evade detection on PC based OS like Windows system, furthermore, an advanced mobile malware has also been discovered targeting the Android platform [3-5].

Therefore, it is necessary to dynamically analyze the newly issued attack mechanism such as remote server-side polymorphic malicious mobile apps, and analyses the indigenous characteristics of polymorphic malicious mobile apps using proposed detection engine. In detail, we implemented a dynamic analysis system for detecting remote server-side polymorphic malicious apps, which has recently become an issue on Android based smartphone.

2. Remote Server-Side Polymorphic Mobile Apps

Android's increase in popularity and its openness have triggered a great rise in malware-spreading apps via the Android Open Market. Android malware apps contain spyware and Trojans as a hidden code shown in [3]. This malicious code is designed to modify itself every time after it's downloaded to make detection more difficult. The code is capable of modifying itself on download in three different ways: (1) Variable data changes, (2) file re-ordering and (3) insertion of dummy files. In more detailed example, we can see the execution patterns of polymorphic malware as follow Figure 1 (com_alioth_imdevil_jp_DevilsCreed_full_1_8.apk). Therefore, it is very hard to detect these kinds of server-side polymorphic (SSP) mobile malware.



Do you agree with the terms of the download com.alioth.imdevil_jp.DevilsCreed.full_1.8 ? To continue the download, click Install

Do you agree with the terms of the download com.alioth.imdevil_jp.DevilsCreed.full_1.8 ? To continue the download, click Install

Unfortunately, Install has stopped.

Figure 1. Remote Server-Side Polymorphic Mobile Apps

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.

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 contrapositives. This type of malware can still be recognized by anti-virus software fairly easily, however, because the decrypted virus body remains the same.

3. Detailed Mechanism of Remote Server-Side Polymorphic Mobile Apps

Remote server-side polymorphic 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 encrypted virus body as a follow-up figure. The mutation engine disassembles the code and represents it with a meta-language that characterizes the code's function but disregards how the code achieves this function. The end result is a new code that bears no resemblance to its original syntax, but is functionally the same. Remote server-side polymorphic malware is more difficult for anti-virus software to recognize, but not impossible. The weakness of metamorphic software is that the mutation engine needs to analyze the code in order to disassemble it and if the mutation engine can analyze the code, so can vendors who make anti-virus software [3, 4].

Most remote SSP codes discovered at present are modified malicious codes of the fake installer (*FakeInst*) [6] and *OpFake* types. With most of these, different types of code values are downloaded every time through a 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 as it shows in Figure 2.

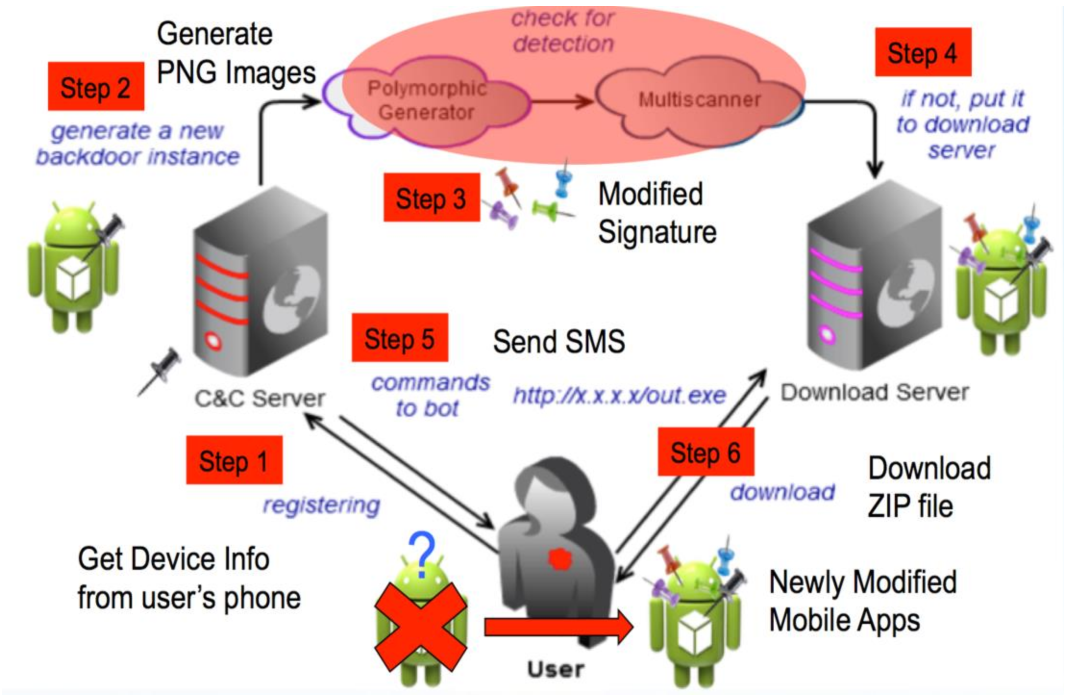
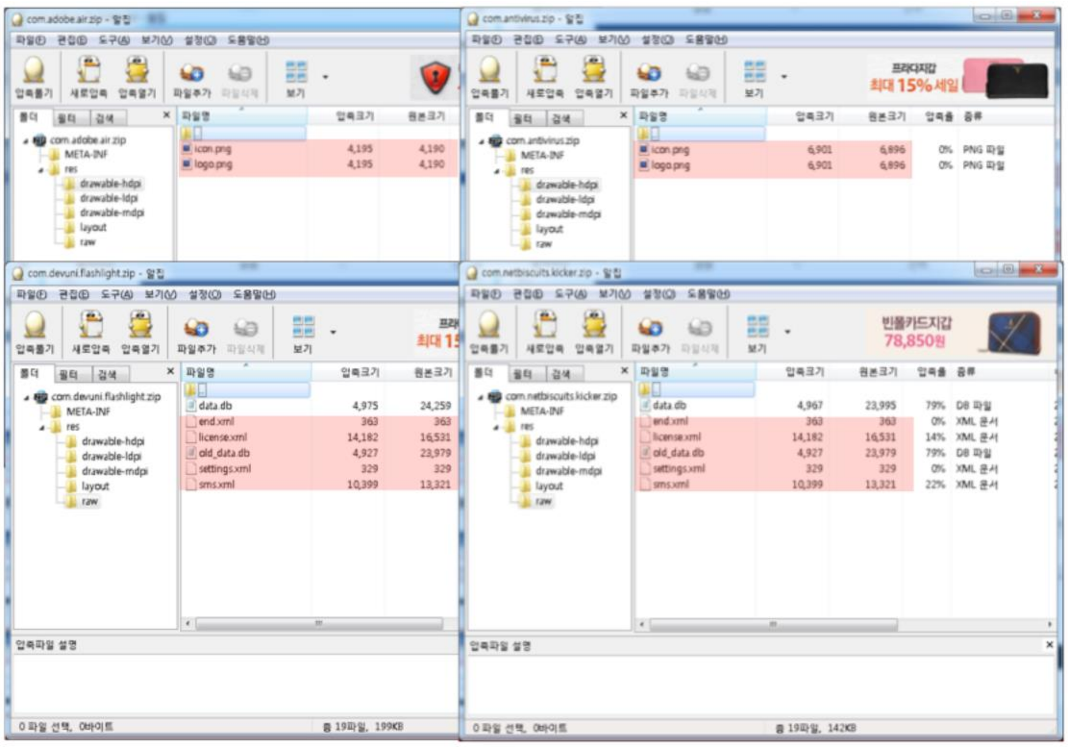


Figure 2. Mechanism of Remote Server-Side Polymorphic Mobile Apps

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.

We compared 4 kinds of Server-Side Polymorphic mobile apps. As a comparison results, we can find that those 4 similar SSP type of malicious mobile apps have an equivalent file size on 3 different files, such as *"AndroidManifest.xml"*, *"classes.dex"* and *"resources.arsc"* files included on each apps respectively! But, they also contain different kinds of icon image as it shows in the figure. For example, both *icon.png* and *logo.png* files embedded on *com.adobe.air.apk* SSP malicious code also can be found on *com.antivirus.apk* SSP malicious code, but the file sizes are not the same as those ones on each other. Therefore, we can calculate different signatures from each SSP malicious mobile apps! Additionally, each SSP apps include different contents of *"data.db"* data as shown on Figure 3. Therefore, each SSP apps derivate different hash & signature value to evade detection from existing mobile vaccines!



com.adobe.air.zip - 알집

com.adobe.air.zip
META-INF
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
raw

파일명	압축크기	원본크기
icon.png	4,195	4,190
logo.png	4,195	4,190

com.antivirus.zip - 알집

com.antivirus.zip
META-INF
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
raw

파일명	압축크기	원본크기	압축률	종류
icon.png	6,901	6,896	0%	PNG 파일
logo.png	6,901	6,896	0%	PNG 파일

com.devuni.flashlight.zip - 알집

com.devuni.flashlight.zip
META-INF
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
raw

파일명	압축크기	원본크기
data.db	4,975	24,259
end.xml	363	363
license.xml	14,182	16,531
old_data.db	4,927	23,979
settings.xml	329	329
sms.xml	10,399	13,321

0 파일 선택, 0바이트　　총 19파일, 199KB

com.netbiscuits.kicker.zip - 알집

com.netbiscuits.kicker.zip
META-INF
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
raw

파일명	압축크기	원본크기	압축률	종류
data.db	4,967	23,995	79%	DB 파일
end.xml	363	363	0%	XML 문서
license.xml	14,182	16,531	14%	XML 문서
old_data.db	4,927	23,979	79%	DB 파일
settings.xml	329	329	0%	XML 문서
sms.xml	10,399	13,321	22%	XML 문서

0 파일 선택, 0바이트　　총 19파일, 142KB

Figure 3. Internal File Analysis Embedded on Remote SSP Mobile Apps

4. Dynamic Analysis System for Detecting Remote Server-Side Polymorphic Mobile Apps

We need to use dynamic analysis mechanism to detect the latest remote server-side polymorphic mobile apps. On most of these malicious apps, different types of code generated from server-side polymorphic server are downloaded every time through a linkage with a remote server. To detect server-side polymorphic malicious mobile apps dynamically, we constructed detection system as shown on Figure 4. At first, we have to upload or submit suspicious mobile apps on PHP based detection server. And then those apps will be sent to python based dynamic analysis module. On this core engine, we can extract the characteristics of the suspicious object and we can get dynamic analysis results on the test sample apps. And those results are stored on DB implemented by MySQL. Finally, we can validate and certify those dynamic analysis results on main PHP based detection server.

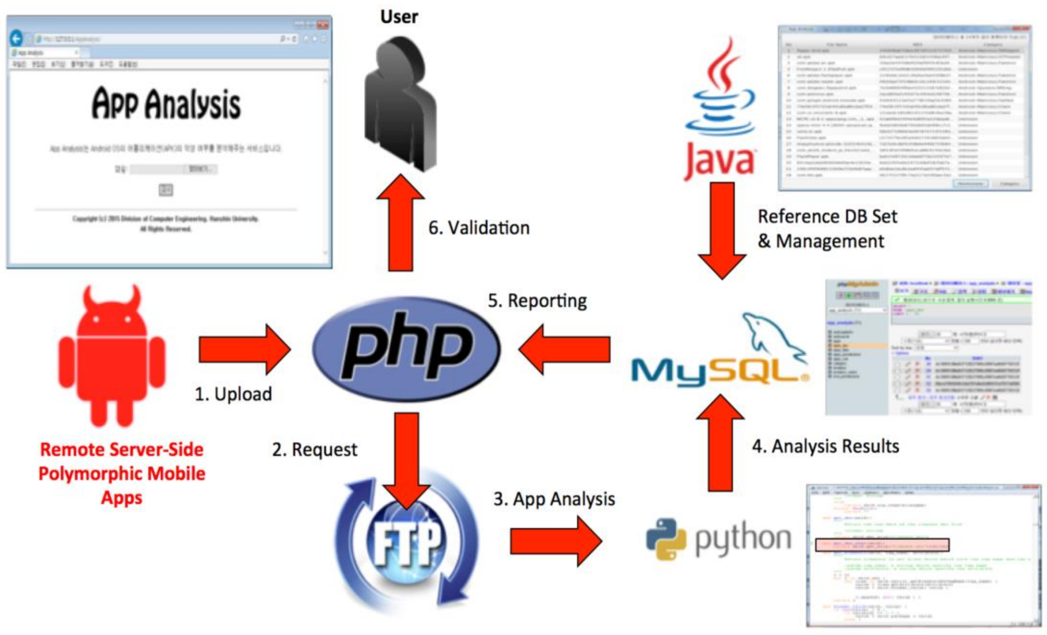


Figure 4. Dynamic Analysis System Architecture for Validating Remote SSP Mobile Apps

In detail, we designed expanded DB schema to manage and store analysis results gathered from suspicious mobile apps as shown on Figure 5. We added more tables such as “app_permissions” and “app_dex” on existing DB tables to manage both permission information and analysis result, which are operated with enhanced dynamic analysis results on proposed “app_analysis” system cooperated with existing *DroidBox* [7, 8] toolkits.

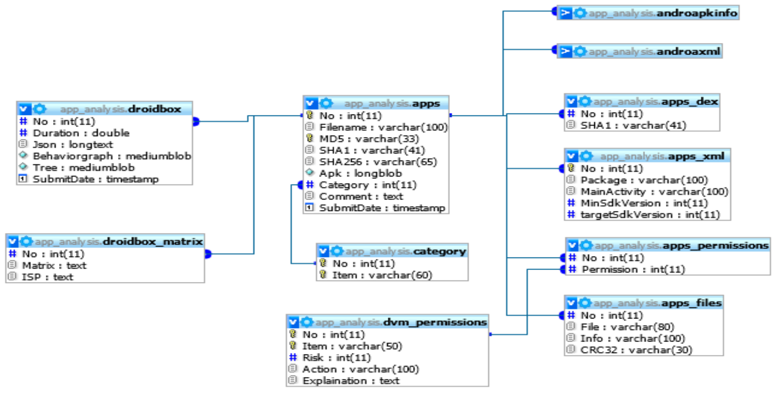
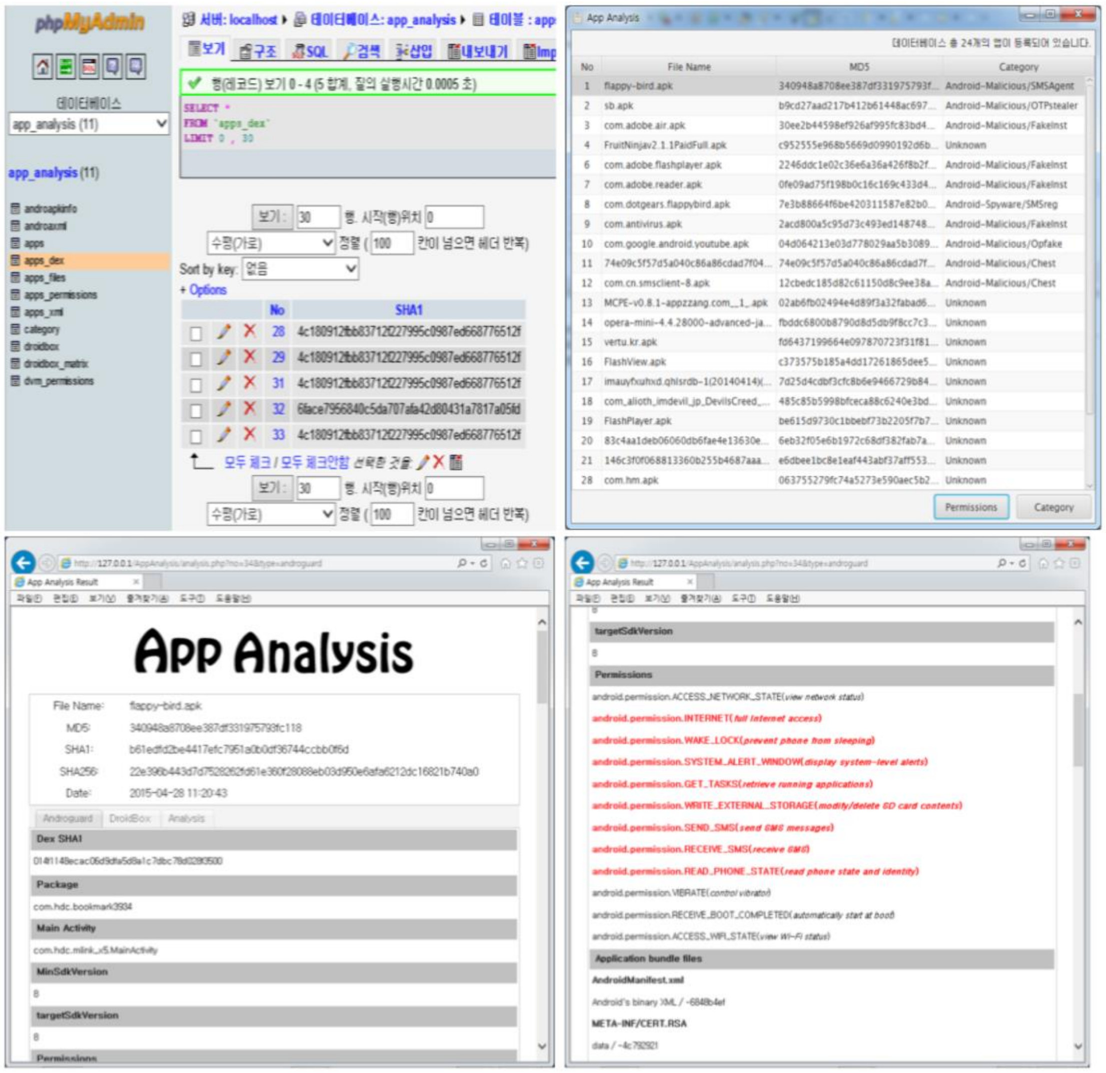


Figure 5. DB Schema for Managing Dynamic Analysis System for Detecting Remote SSP Mobile Apps

5. Implementation of Dynamic Analysis System for SSP

We implemented dynamic analysis system as shown on Figure 6. At first, we provided web based management system, which name is “app_analysis” for maintenance of suspected mobile apps in detail. Proposed system manages detailed internal static analysis result and distinctive characteristics gathered from dynamic analysis procedures in particular. We can find apps core signature that can be used to distinguish remote server-side polymorphic malicious mobile apps from normal ones.



서버: localhost ▸ 데이터베이스: app_analysis ▸ 테이블 : app

행(레코드) 보기 0 - 4 (5 합계, 질의 실행시간 0.0005 초)

```
SELECT *
FROM `apps_dex`
LIMIT 0 , 30
```

데이터베이스: app_analysis (11)

androapkinfo, androaxml, apps, apps_dex, apps_files, apps_permissions, apps_xml, category, droidbox, droidbox_matrix, dvm_permissions

보기 : 30 행. 시작(행)위치 0
수평(가로) 정렬 (100 칸이 넘으면 헤더 반복)
Sort by key: 없음

No	SHA1
28	4c180912fbb837120227995c0987ed668776512f
29	4c180912fbb837120227995c0987ed668776512f
31	4c180912fbb837120227995c0987ed668776512f
32	6face7956840c5da707afa42d80431a7817a05fd
33	4c180912fbb837120227995c0987ed668776512f

모두 체크 / 모두 체크안함 선택한 것을

App Analysis

데이터베이스 총 24개의 앱이 등록되어 있습니다.

No	File Name	MD5	Category
1	flappy-bird.apk	340948a8708ee387df331975793f...	Android-Malicious/SMSAgent
2	sb.apk	b9cd27aad217b412b61448ac697...	Android-Malicious/OTPstealer
3	com.adobe.air.apk	30ee2b44598ef926af995fc83bd4...	Android-Malicious/FakeInst
4	FruitNinjav2.1.1PaidFull.apk	c952555e968b5669d0990192d6b...	Unknown
6	com.adobe.flashplayer.apk	2246ddc1e02c36e6a36a426f8b2f...	Android-Malicious/FakeInst
7	com.adobe.reader.apk	0fe09ad75f198b0c16c169c433d4...	Android-Malicious/FakeInst
8	com.dotgears.flappybird.apk	7e3b88664f6be420311587e82b0...	Android-Spyware/SMSreg
9	com.antivirus.apk	2acd800a5c95d73c493ed148748...	Android-Malicious/FakeInst
10	com.google.android.youtube.apk	04d064213e03d778029aa5b3089...	Android-Malicious/Opfake
11	74e09c5f57d5a040c86a86cdad7f04...	74e09c5f57d5a040c86a86cdad7f...	Android-Malicious/Chest
12	com.cn.smsclient-8.apk	12cbedc185d82c61150d8c9ee38a...	Android-Malicious/Chest
13	MCPE-v0.8.1-appzzang.com__1_.apk	02ab6fb02494e4d89f3a32fabad6...	Unknown
14	opera-mini-4.4.28000-advanced-ja...	fbddc6800b8790d8d5db9f8cc7c3...	Unknown
15	vertu.kr.apk	fd6437199664e097870723f31f81...	Unknown
16	FlashView.apk	c373575b185a4dd17261865dee5...	Unknown
17	imauyfxuhxd.qhisrdb-1(20140414)(...	7d25d4cdbf3cfc8b6e9466729b84...	Unknown
18	com_alioth_imdevil_jp_DevilsCreed_...	485c85b5998bfceca88c6240e3bd...	Unknown
19	FlashPlayer.apk	be615d9730c1bbebf73b2205f7b7...	Unknown
20	83c4aa1deb06060db6fae4e13630e...	6eb32f05e6b1972c68df382fab7a...	Unknown
21	146c3f0f068813360b255b4687aaa...	e6dbee1bc8e1eaf443abf37aff553...	Unknown
28	com.hm.apk	063755279fc74a5273e590aec5b2...	Unknown

http://127.0.0.1/AppAnalysis/analysis.php?no=34&type=androguard

App Analysis Result

App Analysis

File Name:	flappy-bird.apk
MD5:	340948a8708ee387df331975793fc118
SHA1:	b61edfd2be4417efc7951a0b0df36744ccbb0f6d
SHA256:	22e396b443d7d75282f2fd61e360f28088eb03d950e6afa6212dc16821b740a0
Date:	2015-04-28 11:20:43

Androguard | DroidBox | Analysis

Dex SHA1
0141148ecac06d9dfa5d8a1c7dbc78d0280500

Package
com.hdc.bookmark3934

Main Activity
com.hdc.mlink_x5.MainActivity

MinSdkVersion
8

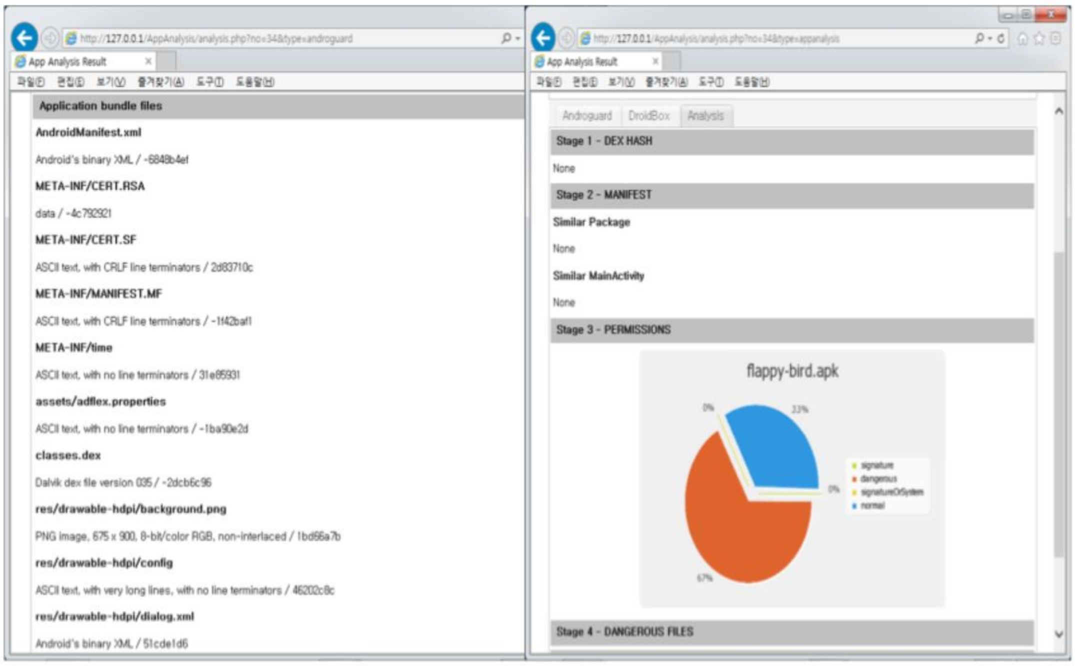
targetSdkVersion
8

Permissions
android.permission.ACCESS_NETWORK_STATE(*view network status*)
android.permission.INTERNET(*full Internet access*)
android.permission.WAKE_LOCK(*prevent phone from sleeping*)
android.permission.SYSTEM_ALERT_WINDOW(*display system-level alerts*)
android.permission.GET_TASKS(*retrieve running applications*)
android.permission.WRITE_EXTERNAL_STORAGE(*modify/delete SD card contents*)
android.permission.SEND_SMS(*send SMS messages*)
android.permission.RECEIVE_SMS(*receive SMS*)
android.permission.READ_PHONE_STATE(*read phone state and identity*)
android.permission.VIBRATE(*control vibrator*)
android.permission.RECEIVE_BOOT_COMPLETED(*automatically start at boot*)
android.permission.ACCESS_WIFI_STATE(*view Wi-Fi status*)

Application bundle files
AndroidManifest.xml
Android's binary XML / -6848b4ef
META-INF/CERT.RSA
data / -4c792921

Figure 6. Implementation of Dynamic Analysis System for Detecting Remote SSP Mobile Apps

By carrying out a static analysis on the subject app, the permission information included in the app could be automatically extracted. Using a proposed system, it was also possible to find an illegal access module included within the terminal internal file by analyzing the internal remote SSP code such as *com.adobe.air.apk*. Additionally, it was possible to obtain the internal codes of remote SSP malicious apps; thus, modules or functions with suspected abnormality were automatically extracted. By using the integrated dynamic analysis mechanism, it was possible to carry out an analysis and differentiation process of polymorphic malicious apps collected online. We modified the core functions of the pre-existing *DroidBox* module and primarily establish a group of data on metamorphic malicious apps through establishing DB. And then we also provided the additional feature of enhancing accuracy on detecting SSP malicious apps using the established DB information. The results of static and dynamic analysis could be presented in an integrated form along with the permission information included within the app as shown on Figure 7.


http://127.0.0.1/AppAnalysis/analysis.php?no=34&type=androguard
App Analysis Result
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

Application bundle files

AndroidManifest.xml
Android's binary XML / -6848b4ef
META-INF/CERT.RSA
data / -4c792921
META-INF/CERT.SF
ASCII text, with CRLF line terminators / 2d83710c
META-INF/MANIFEST.MF
ASCII text, with CRLF line terminators / -1f42baf1
META-INF/time
ASCII text, with no line terminators / 31e85931
assets/adflex.properties
ASCII text, with no line terminators / -1ba90e2d
classes.dex
Dalvik dex file version 035 / -2dcb6c96
res/drawable-hdpi/background.png
PNG image, 675 x 900, 8-bit/color RGB, non-interlaced / 1bd66a7b
res/drawable-hdpi/config
ASCII text, with very long lines, with no line terminators / 46202c8c
res/drawable-hdpi/dialog.xml
Android's binary XML / 51cde1d6

http://127.0.0.1/AppAnalysis/analysis.php?no=34&type=appanalysis
App Analysis Result
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
Androguard
DroidBox
Analysis

Stage 1 - DEX HASH
None
Stage 2 - MANIFEST
Similar Package
None
Similar MainActivity
None
Stage 3 - PERMISSIONS

Stage 4 - DANGEROUS FILES

Figure 7. Web based Dynamic Analysis system against Server-Side Polymorphic Mobile Apps

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 Remote Server-side Polymorphic form. This method ultimately identified it as a polymorphic malicious app. In addition, on this basis of information from the behavior graphs, an underlying execution pattern in all polymorphic malicious apps was found.

6. Conclusions

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 server-side polymorphic 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 provide a method for effective detection of suspicious malicious apps. Therefore, we implemented dynamic analysis system to manage and classify remote SSP type of malicious apps from normal ones. Using a proposed system, user can validate and certify whether suspicious mobile apps are malicious or not using aggregated DB data set connected by web interface.

Acknowledgment

This research was 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

[1] D. He, “Mobile application security: malware threats and defenses”, Wireless Communications, IEEE, vol. 22, no.1, **(2015)**.
[2] Server-side Polymorphic Android Applications, http://www.symantec.com/connect/blogs/server-side-polymorphic-android-applications
[3] Server-side polymorphism: How mutating web malware tries to defeat anti-virus software, http://nakedsecurity.sophos.com/2012/07/31/server-side-polymorphism-malware/
[4] Detecting Polymorphic Malware, http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/ detecting -polymorphic-malware
[5] What you Need to Know About Android Trojans, http://pocketnow.com/2013/04/03/android-trojans
[6] ‘FakeInstaller’ Leads the Attack on Android Phones, http://blogs.mcafee.com/mcafee-labs/fakeinstaller-leads-the-attack-on-android-phones.
[7] H. Sun, “A programming model and framework for comprehensive dynamic analysis on Android”, Modularity 2015, **(2015)**, pp.133-145.
[8] “Android Dynamic Code Analysis – Mastering DroidBox”, http://blog.dornea.nu/2014/08/05/android-dynamic-code-analysis-mastering-droidbox/
[9] W. Zhou, Y. Zhou, X. Jiang and P. Ning, “DroidMOSS: Detecting Repackaged Smartphone Applications in Third- Party Android Marketplaces”, Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy, **(2012)**.
[10] A. Jonathan, P. Marpaung, N. Bruce and J. L. Hoon, “Higher-Order Countermeasure against Side-Channel Cryptoanalysis on Rabbit Stream Cipher”, Journal of information and communication convergence engineering, vol. 12, no. 4, **(2014)**, pp. 237-245.
[11] H. S. Lee and H.-W. Lee, “Dynamic Analysis of Remote Server-Side Polymorphic Malicious Mobile Apps on Android based Smartphone”, 2015 International Conference on Future Information & Communication Engineering (ICFICE2015), **(2015)**.

Authors

Han Seong Lee, he is currently an undergraduate student at the Division of Computer Engineering, Hanshin University, Gyyeong-gi Province, Korea. His research interest is in the areas of Android Smart Devices, Network and Java Programming for Computer Security & Digital Forensics.

Hyung-Woo Lee, he received B.S, M.S and Ph.D degrees 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.