International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

A Study on Security Issues Caused by Ad Libraries in Smartphone Apps

Ming-Yang Su, Meng-Yuan Chang, and Ying-Tang Hsu

***Abstract*—Many app developers would cooperate with ad networks by adding a procedural code (called ad library or ad lib) from ad providers into the app. The free apps will help them get into the market, and the ad will increase their income. To make ad more targeted at the needs of individual users, ad lib must collect personal information, including age, gender, income and something even more private, the habit of network use, and the location of users. It would automatically connect itself with the ad server to send out the personal information while receiving the ad reply from the ad server. Aside from pictures, it may include JavaScript for malicious acts and require the connection between the app and the third-party server to download malicious apps and automatically install and run them under the background. To make more profits from ads, many app developers embed more than one ad libs into an app, i.e., connecting to several ad providers, so as to make the security problem ever worse. Using the machine learning on the static program analysis, this paper developed an anti-ad app called *Ad Pioneer*. Users can adopt *Ad Pioneer* to check the security of an app's ad, so as to prevent personal information leaking.**

***Keywords*—Smartphone, Ad SDK (ad lib), Ad Server, Ad Request, Ad Reply, Privacy, Permission, Sensitive Function**

I. Introduction

App ad is greatly different from browser ad, for the ad lib of the former is compiled after being added into the app. In other words, advertising has been regarded as one of normal functions of the app in the system. Hence, the ad lib can use the permission of the whole app. As for the latter, browser and ad are seen as different processes in the system and thus have different permissions.

Android is an open and highly popular operating system. In 2014, its software exceeded 1.4 million in number [1]; in 2015 the quantity of the APPs on its application software store Google Play surpassed that on the APPLE App Store, which made it an operating system with the largest number of apps in the world. Many Android apps are free, but app developers make profits through advertising. To make advertising more targeted at the needs of individual users, ad lib embedded into the app must collect some personal information, including age, gender and income. Many ad libs may break the permission to collect private information about cell phone users and their habits of network use or track their locations. Without affecting users' operation of the app, ad lib automatically connects itself with the ad server and send out the collected personal information through the ad request packages. Meanwhile, it receives the ad reply packages from the ad server. Apart from ad pictures, it may include JavaScript and take some actions malicious to users without any authorization, such as collecting the information about the address book, audios and image documents, sending short message and emails, or steal the cookies of cell phone users, so as to obtain the accounts and codes of users. Ad reply may also request the connection between the app and the third-party server to download malicious apps and automatically install and run them under the background.

The ad libs embedded into the app would obtain contents from ad providers and display them on the user interface of the app [2]. According to the latest findings of the cooperation between Purdue University and Microsoft [3], although the App dependent on ads runs under the banner of "free", it costs power – the most important basis of smart phones and even other electronic devices. According to the report, about 75% of the power of the free apps in the Android system is used on ad service, tracking and uploading of information about users. In an extreme situation, a free app alone may consume all the power of a device within 90 minutes. Abhinav Pathak, a researcher at Purdue University, used an HTC cell phone equipped with Android 2.3 to test 5 popular apps in Android, including the famous Angry Birds. In the test on Angry Bird, only 20% of the power was used for running the game; 45%, uploading the location of users in the GPS and delivering relevant ads through the 3G network. Even if the information transmission is finished, the 3G network did not stop working; instead, it continued to consume 28% of power. Additionally, power is not consumed by App ads alone. In the test on the App and the browser version of *New York Times*, both consumed 15% of power to transmit the information about users.

Ming-Yang Su
Department of Computer Science and Information Engineering,
Ming Chuan University,
Taiwan
minysu@mail.mcu.edu.tw

Meng-Yuan Chang
Department of Computer Science and Information Engineering,
Ming Chuan University,
Taiwan
a9010726@gmail.com

Ying-Tang Hsu
Department of Computer Science and Information Engineering,
Ming Chuan University,
Taiwan
wen591021@gmail.com

There are many famous Ad Networks, such as Admob, Airpush, Appenda, LeadBolt, Moolah Media and Startapp, as is shown in Figure 1 [3]. Different Ad Networks have different ad libs, and there are both beneficial and malicious ad libs. The most trustworthy one is the ad lib of Admob, a network advertiser acquired by Google. As a highly reliable ad lib, it is also the most widely-used one in free Apps. In contrast, the most criticized one is Airpush, for the annoying ad lib delivers overwhelming ads in the notice column of cell phones on an irregular basis. Some ad libs are notorious for using too much permission and delivering porn ads. Against such a backdrop, the anti-ad software was created. It collects the feature codes of the ad libs and detects them in the downloaded Apps, working like a piece of anti-virus software. A free App stores and retrieves the information of two networks: on one hand, it stores and retrieves the data about games or apps from App developers; on the other, it collects the information for ad from the ad server and uploads the information about cell phones to ad providers [3].

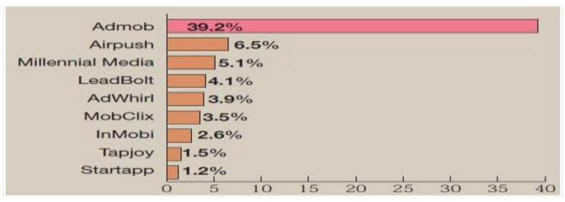


Figure 1. Top 10 Ad Libs of the Free Apps in Android [3]

The delivery ad on the Android platform is a new network ad. As its delivery does not stop with the suspension of apps and ad may appear in the notice column at any time, it is often regarded as spam. In fact, the delivery ad also causes other negative impacts, including the soaring expenses for mobile data, the increase in power consumption, the narrowing available space of cell phones, and the vulnerability to more malicious apps. Pearce et al. [11] changed the Android architecture and proposed a new ad API, matching ad libs with corresponding exclusive ad permissions. This method greatly improved the control over the ad in the Android system. However, the systematic effect can only be achieved through the official re-writing the inner architecture of the Android system and the consensus with the whole advertise ecosystem. Consequently, the method is not highly feasible. In Section 2, the concept of the static program analysis of advertising is introduced; Section 3 illustrates the architecture of this paper as well as the results of the experiment; in Section 4, a short conclusion is presented.

II. Program Analysis of Advertising

According to Kumar and Singh [12], many ad libs request excessive permission or use permission without authorization. Some apps even sniff network traffic to obtain the ad request packages which involves several ad providers, so as to obtain the personal information about users on an extensive basis. This paper also focuses on how some notorious ad libs transmit personal information to unknown servers through the third-party connection. According to Gao et al. [13], ad lib and app are compiled after combination, so it is impossible to prevent ad lib from using the permission without authorization. The author has designed a system called PmDroid to prevent the information obtained without permission by ad lib from being sent to ad servers. PmDroid shows the severity of the excessive permission of ad libs on a graphic interface. To deepen the understanding of ad lib, the author made 53 apps and each app was embedded with a different ad lib. All these apps did nothing but announced the permission of all Android systems. The packaged traffic of all the apps was recorded to get the information about the abuse of permission of ad libs. As the apps did nothing, all the traffic was consumed by the ad libs. The author came to the conclusion that the abuse of permission of ad libs is really serious.

Static program analysis means that an app is analyzed when it is not being used [4]. The analysis focuses on understanding app behaviors through analyzing the original app codes or disassembly. But in the actual analysis, actual implementation software is used to improve the understanding of the results of static program analysis. Starting with static program analysis, Schmidt et al. [5] proposed to use readelf to extract the function call list of the Executable and Linking Format (ELF) archives and employ the classification algorithm to classify the collected samples, so as to detect malicious software. Apvrille and Strazzere put forward another static program analysis [6], where 39 signs like Java API transfer, the existence of embedded executable documents, code size and website were adopted. Each sign was equipped with a weight, and the statistical calculation would show which signs were most frequently used in the app codes of the developers of malicious apps for Android. Under the Android environment, the subject of static program analysis is the Android Application Package File (APK) [7]. The APK structure comprises five parts: 1) the META-INF file, 2) the res file. 3) AndroidManifest.xml, 4) class.dex and 5) resources.arsc. As far as the five parts are concerned, Fuchs et al. [8] proposed to use ScanDroid to collect the permission required by the software and then find out if the software was consistent with the permission according to the data flow of the software. The problem of the method is that it cannot detect the malicious software which purposely bypasses the Android permission mechanism. Our analysis, however, focuses on class.dex. After obtaining the original app code through the anti-compilation of the archive, we can find out all sensitive function in the original codes for the classification. For example:

- ActivityManager\;\-\>killBackgroundPr ocesses: suspend the process
- TelephonyManager\;\-\>getDeviceId”: collect the IMEI codes, phone numbers and system versions of cell phones
- SmsReceiver\;\-\>abortBroadcast”:

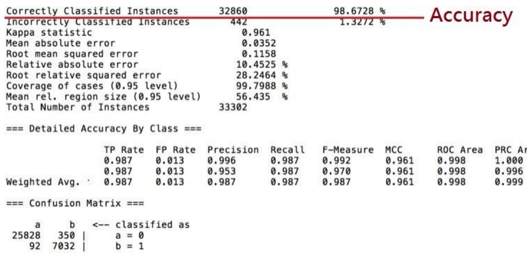
interrupt the receiving of short message

- Chmod: alter the document permission
- HttpClient\;\-\>execute: send a request to the far-end server
- ContentResolver\;\-\>query: read the information about contact persons and short message

We collected many pieces of malicious software as well as analyzed and summarized the behaviors of a large number of malicious apps. Then, we found out the sensitive functions that might be used in the apps and adopted Weka [7] to establish a feature model.

III. Architecture and Experiment Data

First, we visited the website with the ranking of ad providers [10] and selected the top 50 around the world. The total share held by these ad providers is 95% of the market, so their reliability as the test case was high. The data downloaded from apk download or APK mirror APKPure were used for the Weka model training. Then, all the archives were anti-compiled and excerpted and divided into two groups – the “ad files” and the “non-ad files”. Two selection methods were adopted. The first one was the preset-arrayed automatic scanning of the self-made apps which worked according to the fixed name of the ad files. The second one was the manual selection and scanning which aimed to ensure the accuracy and completeness of the data. There were over ten thousand data in total. After the data were added into the database, they were stored in arff files, and the path column “attributes” was manually changed into “String”. If “attributes” was “Nominal”, many algorithm classifiers would be tried. With the built-in string cutting function of Weka [9], this study trained an evaluation model (strings were cut for a higher accuracy). Three algorithms were analyzed in the experiment, including Support Vector Machine (SVM), ZeroR, and Naive Bayes. Finally, Navie Bayes, the algorithm which had the best performance, was adopted. The accuracy of the 10-fold cross-validation could be up to 98.3665%.



```
Correctly Classified Instances       32860               98.6728 %
Incorrectly Classified Instances       442                1.3272 %
Kappa statistic                          0.961
Mean absolute error                      0.0352
Root mean squared error                  0.1158
Relative absolute error                 10.4525 %
Root relative squared error             28.2464 %
Coverage of cases (0.95 level)          99.7988 %
Mean rel. region size (0.95 level)      56.435  %
Total Number of Instances            33302

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.987    0.013    0.996      0.987    0.992      0.961    0.998     1.000     0
                 0.987    0.013    0.953      0.987    0.970      0.961    0.998     0.996     1
Weighted Avg.    0.987    0.013    0.987      0.987    0.987      0.961    0.998     0.999

=== Confusion Matrix ===

     a     b   <-- classified as
 25828   350 |     a = 0
    92  7032 |     b = 1
```

Figure 2. Accuracy of Navie Bayes

Then, the Weka training model was added into the malicious ad monitoring app (Ad Pioneer) developed in this study. The system adopted the API and Weka API provided by Android SDK, the open original code resources in Github, the anti-compilation, as well as the Weka model “bayes.model” from the self-made ad database. The development was as follows: 1) Made the appearance; 2) made the plane-entering full-screen effect; 3) designed the archive selection interface; 4) adopted Weka API to establish Attribute-Relation File Format(arff); 5) designed Attribute, Instance, and Value in arff; 6) used bayes.model to conduct the predicted experiment on the established arff; 7) Decompressed apk; 8) adopted jaDX and dex2jar for the anti-compilation; 9) added the names of the files into arff and made the prediction; 10) made the sensitive function analysis of the app codes after defining the ad files; 11) analyzed the scores and informed users of the damage to privacy. “Ad Pioneer” is shown in Figures 3 and 4. It is an App created in the Android Studio development environment. Currently, its functions are as follows: browse the archives in cell phones and monitor the APK archive; use the feature model established in the special case to evaluate the danger and recognize malicious ads; list the malicious acts on the backstage.

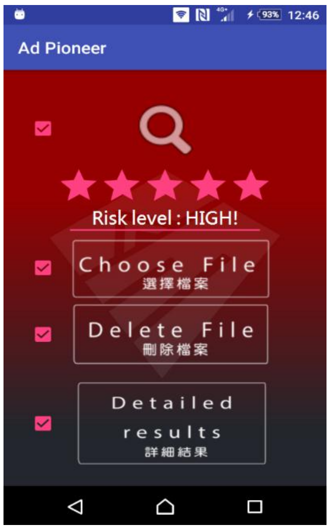
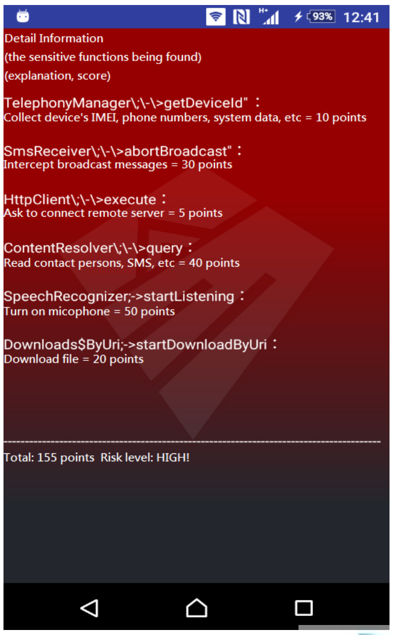


Figure 3. Menu of Ad Pioneer



12:41

Detail Information
(the sensitive functions being found)
(explanation, score)

TelephonyManager\;\-\>getDeviceId" :
Collect device's IMEI, phone numbers, system data, etc = 10 points

SmsReceiver\;\-\>abortBroadcast" :
Intercept broadcast messages = 30 points

HttpClient\;\-\>execute :
Ask to connect remote server = 5 points

ContentResolver\;\-\>query :
Read contact persons, SMS, etc = 40 points

SpeechRecognizer;->startListening :
Turn on micophone = 50 points

Downloads$ByUri;->startDownloadByUri :
Download file = 20 points

Total: 155 points Risk level: HIGH!

Figure 4. List of Detailed Results

International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

IV. Conclusion

In this paper, static program analysis was adopted to probe into the monitoring of the mainstream malicious ads in the Android system. Weka and diverse static program analyses were used to obtain satisfactory results in the monitoring of malicious ads. Aside from detecting malicious APPs with the Google Play built in Android, users can also install the models which can monitor features or adopt anti-virus APPS which can detect some unknown malicious software (such as MAB) to prevent the problems causing the leakage of personal information and abnormalities. As far as the current experiment is concerned, Naive Bayes and the 10-fold cross-validation can contribute to an accuracy of up to 98.3665%.

Acknowledgment

This work was partially supported by the Ministry of Science and Technology with contract MOST 104-2221-E-130-008 and 105-2221-E-130-004.

References

[1] NaoyaKajiwara, Shinichi Matsumoto,YuukiNishimoto, Yoshiaki Hori, Kouichi Sakurai,.2014, “Detection of Privacy SensitiveInformation Retrieval Using API Call LoggingMechanism within Android Framework,” Journal of Networks, Vol 9, No 11 (2014), 2905-2913,ov 2014.
[2] J Crussell, R Stevens, H Chen, Madfraud: Investigating ad fraud in android applications, Proceedings of the 12th annual international conference on Mobile systems, applications, and services, Pages 123-134, June 16 - 19, 2014.
[3] http://blog.trendmicro.com.tw/?p=9022#more-9022
[4]Wikipedia.,2015,APK.(http://en.wikipedia.org/wiki/Android_application_package)
[5]Aubrey-Derrick Schmidt et al., “Static analysis of executables for collaborative malware detection on Android,” Proceedings of the IEEE International Conference on Communications, pp. 631-635, 2009.
[6] A. Apvrille and T. Strazzere, "Reducing the window of opportunity for Android malware gotta catch 'em all," Journal in Computer Virology, vol.8,pp.61-71,2012.
[7] Weka, http://www.cs.waikato.ac.nz/ml/weka/
[8] Adam P. Fuchs, AvikChaudhuri, and Jeffrey S.Foster. Technical Report CS-TR-4991,Department of Computer Science, University of Maryland, College Park, November 2009. 15.Wu Zhou, Yajin Zhou, Xuxian Jiang,PengNing, "DroidMOSS: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces," Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY 2012), San Antonio, TX, February 2012.
[9]http://zh.wikipedia.org/wiki/%E6%94%AF%E6%8C%81%E5%90%91%E9%87%8F%E6%9C%BA
[10] http://www.appbrain.com/stats/libraries/ad?list=top500
[11] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner, “AdDroid: Privilege Separation for Applications and Advertisers in Android,” Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, 2012.
[12] Pradeep Kumar and Maninder Singh, “Mobile Applications: Analyzing Private Data Leakage Using Third Party Connections,” in the IEEE Proceedings of International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 57-62, 2015.
[13] Xing Gao, Dachuan Liu, Haining Wang, and Kun Sun, “PmDroid: Permission Supervision for Android Advertising,” in the IEEE Proceedings of the 34th Symposium on Reliable Distributed Systems, pp.120-129, 2015.

Ming-Yang Su received his B.S. degree from the Department of Computer Science and Information Engineering of Tunghai University, Taiwan in 1989, and received his M.S. and Ph.D. degrees from the same department of the National Central University and National Taiwan University in 1991 and 1997, respectively. He is an IEEE member, and currently a professor of the Department of Computer Science and Information Engineering at the Ming Chuan University, Taoyuan, Taiwan. His research interests include network security, intrusion detection/prevention, malware detection, mobile ad hoc networks, VoIP security and wireless sensor networks.

Meng-Yuan Chang is currently a student of the Department of Computer Science and Information Engineering of Ming Chuan University, Taoyuan, Taiwan. His research interests are in the areas of network security, and mobile phone security.

Ying-Tang Hsu is currently a student of the Department of Computer Science and Information Engineering of Ming Chuan University, Taoyuan, Taiwan. His research interests are in the areas of intrusion detection/prevention, wireless sensor networks and malware detection.