Detection of Android Ad Library Focusing on HTTP Connections and View Object Redraw Behaviors

Kajiwara, Naoya
Department of Informatics, Kyushu University

Kawamoto, Junpei
Department of Informatics, Kyushu University

Matsumoto, Shinichi
Department of Informatics, Kyushu University

Hori, Yoshiaki
Organization for General Education, Saga University

他

出版情報：Proceedings of the 29th International Conference on Information Networking, pp.104-109, 2015. IEEE Computer Society
バージョン：accepted
権利関係：

Detection of Android Ad Library Focusing on HTTP Connections and View Object Redraw Behaviors

Naoya Kajiwara
Department of Informatics,
Kyushu University,
ISIT,
Fukuoka, Japan
Email: kajiwara@itslab.inf.kyushu-u.ac.jp

Junpei Kawamoto
Department of Informatics,
Kyushu University,
ISIT,
Fukuoka, Japan
Email:kawamoto@itslab.inf.kyushu-u.ac.jp

Shinichi Matsumoto
Department of Informatics,
Kyushu University,
ISIT,
Fukuoka, Japan
Email:smatsumoto@isit.or.jp

Yoshiaki Hori
Organization for General Education,
Saga University,
ISIT,
Saga, Japan
Email:horiyo@cc.saga-u.ac.jp

Kouichi Sakurai
Department of Informatics,
Kyushu University,
ISIT,
Fukuoka, Japan
Email:sakurai@csce.kyushu-u.ac.jp

***Abstract*—In recent years, the smart phone application market has expanded rapidly. One of reasons is the popularity of free applications. A developer acquires his revenues by including advertising libraries in his own application. However, some problems about these advertising libraries become clear from recent researches. Especially in the leakage of privacy information is known as a typical problem which advertising libraries cause. In order to solve this problem, the technology which detects advertisement libraries is important. In this paper, we propose a method for detection of Android ad library. We focus on the acquisition and redraw of advertising image operation which are the basic operations of mobile advertisement. Firstly, we tried running some applications with advertisements. Then, It turned out that mobile advertisements acquire advertising images from server and set that image on the screen at a fixed interval. By modifying AndroidOS, logging HTTP connections and View object redraw behaviors, we confirmed the ad image acquisition behavior. Moreover, to take advantage of the periodicity of this behavior, we carried out Fourier-transform the invocation time data of HTTP connections and redraw of View objects. Then, we extracted the periodicity by calculating correlation coefficient for these two data. From the value of correlation coefficient, it is possible to judge whether advertisement library is incorporated into an application or not. As a result, our proposal method results in a output of about 76 % detection rate.**

I. Introduction

Now, more than 1 300 000 Android applications are exhibited on Google Play market[1]. About 84 % applications of these applications is exhibited for free. Many of the developers of such free applications are selecting monetization model of acquiring the counter value of development by incorporating advertisement into their application[2]. However, it becomes clear about some problems which the advertising libraries have.

The leakage of privacy information is a typical problem caused by advertising libraries. This problem is that when an application which an advertisement library is incorporated into is run, that advertisement library send user's information outside from terminal without acquiring user's consent. For example, some advertising libraries access user's location information, and send it external server[3]. In the background of such problems, there are a fact that developers of applications and advertising libraries are different. About the behavior of an advertising library, an application developer can grasp to some extent from the document. However, application developers have few understanding to the behavior of advertising libraries because of some reasons such as the insufficient document or the low concerns about privacy problem. This means that application developers incorporate third party library whose behavior is unknown. Thus, since application developers do not grasp the behaviors of advertisement correctly, unexpected leakage of information is caused. Moreover, this leads to the problem that developer's explanation of the application in Google Play market will not function. Thus, for both application developers and users, the present condition is that many applications whose behaviors are unknown are in circulation. Therefore, the method of detecting advertisement in Android applications in advance is important.

In this paper, we propose a method for detection of Android ad library. We focus on the acquisition and redraw of advertising image operation, which are the basic operations of mobile advertisement. Firstly, we tried running some applications with advertisements. Then, It turned out that mobile advertisements acquire advertising images from server and set that image on the screen at a fixed interval. By modifying AndroidOS, logging HTTP connections and View object redraw behaviors, we confirmed ad image acquisition behavior. Moreover, to

This work was supported by JSPS KAKENHI Grant Number 26330169.

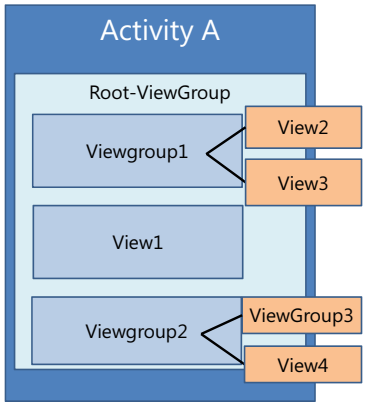

Fig. 1. View Tree Structure on an Activity

take advantage of periodicity of this behavior, we carried out Fourier-transform for the invocation time data of HTTP connections and redraw of View objects. Then, we realized analysis with the tolerance to a noise by analyzing data in a frequency domain. As a result, our proposal method results in a output of about 76 % detection rate.

II. Background

A. Advertisement Library

Generally, the advertisement for smart phones monopolizes a part of screen space, and displays an advertising image there. If a user clicks on an advertising image, actions such as a guidance to an application store or a special web page is performed.
Advertisement library is incorporated into the same package with application's code. Therefore, in Android systems, application code and advertising code share same process ID. This makes distinction of application's function invocation and advertising function invocation difficult.

B. Activity

Activity is a component equivalent to a screen of an Android application. Mostly, Activity and a screen of UI has a relation of 1 to 1. In other words, an application's screens are implemented by Activity classes.

C. View

View is a component equivalent to elements of the screen in an Android application. All of the elements arranged on the screen such as a button, text box or list are implemented by View class or View's child classes. These View objects are set to Activity object corresponding to the screen where a developer wants to set a View in UI. At this time, the View objects take a tree structure. Fig.1 shows an example of View tree structure on an Activity.

III. Related Works

There are some researches related to Android advertisement.
[4] is a paper which proposes detection method of Android advertising library. In this paper, authors focus on acquisition of advertisement image by advertising library in the same way as this paper. By capturing HTTP communications for an acquisition of advertising image and generating graphs from HTTP sessions data, authors detected image acquisition behavior by advertisements. However, there is a challenge that if communication is encrypted, their method cannot be adapted for.
[5] is a paper which proposes the solution method of the permission sharing problem. There is a problem that advertising libraries share Permissions with the applications code. This is because applications code and advertising libraries are implemented into same package and they run on the same process in an Android system. In order to solve this problem, authors proposed the separation of Permission between application's code and advertising library. This system is achieved by running special system service managing advertising functions. Thereby, the check of whether the advertisement is contained in an application become easy, and Permissions can be managed strictly.
[6] is a paper which proposes a testing method of Android application based on UI elements. In this paper authors focus on the Activity and View objects which compose an application's screen. By exploring all activity and view objects, SmartDroid can automatically detect UI-based trigger conditions which leads to the sensitive behavior caused by Android malwares. As a result, authors show that SmartDroid can detect several Android malwares which cannot be detected with existing techniques.

IV. Approach

In this section, we describe the approach about the advertisement detection method. In this paper, we focus on an updating ad image behavior which is the basic operation of the Android ad libraries.

A. Assumption about Advertisement Behavior

Generally, an ad library changes images of the advertisement displayed on a screen for every definition period to other pictures to pull a user's attention. In this paper, we classified this updating ad image operation to the following two behaviors.

1) Http connection
   Picture acquisition from advertising servers are executed by HTTP connections.
2) View Redraw
   Rearrangement of the acquired advertising pictures to the screen

Then, we designed the system that records these two behaviors' invocation time and judging automatically whether an application has advertisement or not using these two data. As mentioned in Section II-A, application's code and advertising code are executed on the same process, distinction of these two are difficult. However, in our assumption, advertisement's behavior have periodicity and application's behavior don't have periodicity. Therefore, in the detection of advertisements, it is important that updating operations of advertisement pictures are always invoked at a fixed interval once an advertisement is set to a screen. That is, when an advertisement is contained in an application, HTTP and View behavior invocations have

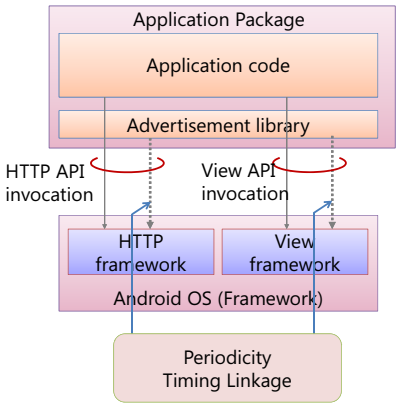

Fig. 2. The Periodicity in Ad-Library's behaviors

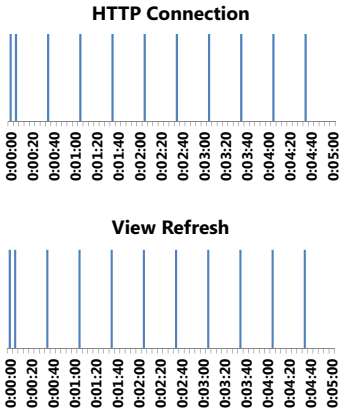

Fig. 3. Invocation Time of HTTP Connection and View Redraw when Running the Application A which has advertisement

a certain periodicity. So, by confirming the existence of this periodicity, it can be judged whether an application has advertisement or not. Fig.2 shows our model of ad behavior's periodicity.

B. Monitoring HTTP connection and View Redraw

We modified Android OS to record invocations of Http connection and View redraw behaviors. We modified the Android framework, Java libraries and apache libraries which provide API for developers. The modified points are "HTTP framework" and "View framework" in Fig. 2. This behavior recording method in Android framework is proposed in [7].

1) Recording HTTP connection: In the document of HTTP connection implementation in Android Developers[8], the use of following two classes is recommended when a developer implements HTTP communication in an Android application.

- *HttpClient* class in org.apache library
- *HttpURLConnection* class in java.net library

However, for the case of a developer implements HTTP connection without using these classes, we inserted codes logging Http connection into

- *Socket.connect(SocketAddress endpoint, int timeout)*

in java.net library. Also when the two above-mentioned classes recommended in Android Developer are used, it is confirmed that a Socket.connect method is invoked inside.

2) Recording View Redraw: Record of the redraw behavior of View was implemented using android.view.ViewTreeObserver class. As mentioned in Section II-C, in AndroidOS, each screen of applications corresponds with Activity object. And, under each Activity, Views are arranged with a tree structure. A ViewTreeObserver class has a function that supervising this View tree structure and detecting events such as an addition of new View or change of View's setting. In this paper, we modified Activity class code in Android Framework, and implemented following operations.

1) At the generation of Activity object, root View object in that Activity is acquired.
2) ViewTreeObserver class is set to root View object.
3) ViewTreeObserver.onGlobalLayoutListener is invoked when View tree state is changed.
4) Log message is recorded within event listener that implies redraw of View was performed.

C. Preliminary experiment

As mentioned in Section IV-A, in our method, we want to detect advertisement in an application by the periodicity of HTTP connection and View redraw behaviors invoked by an application. In this section, we report the result of preliminary experiment. As a preliminary experiment, we ran the following three applications and confirmed that what kind of features appeared about HTTP connection and View redraw behaviors invocation time.

- Application A has an advertisement. This application has no function of HTTP connection and View redraw excepting advertisement.
- Application B has an advertisement. This application has functions of View redraw in addition to advertising function.
- Application C has no advertisement. This application has function of HTTP connection and View redraw in application itself code.

1) Result with Advertisement:

a) Case of Application A: Fig. 3 shows the invocation time of HTTP connection and View redraw behaviors when the application A was run on Android emulator. In this graph, horizontal axis expresses time. The period of graph is 5 minutes which starts from the launch of the application A. And if HTTP connection or View redraw is invoked, vertical bar is appeared on each time slot.
This figure shows that HTTP connections and View redraw behaviors are invoked at intervals of about 30 seconds respectively. Each invocation indicates HTTP communication for acquisition of an advertising image file and an arrangement of an acquired image to the screen.

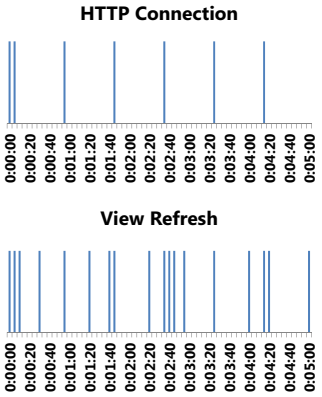

Fig. 4. Invocation Time of HTTP Connection and View Redraw when Running the Application B which has advertisement

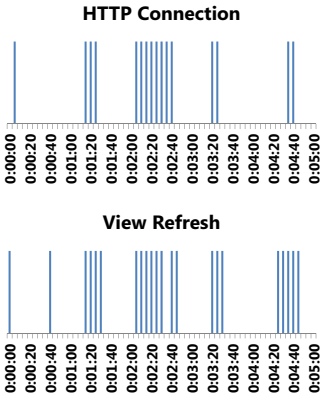

Fig. 5. Invocation Time of HTTP Connection and View Redraw when Running the Application C which has no Advertisement

b) Case of Application B: Fig. 4 shows that invocation time of HTTP connections and View redraw behaviors when the application B with advertisement was run on Android emulator. This figure shows that HTTP connections are executed at the intervals of about 30 seconds. On the other hand, View redraw behaviors data seems to have no periodicity. This is because HTTP connection is not implemented in the code of application itself but View redraw is implemented. As a result, View redraw behaviors by application and advertisement are mixed. In other words, data with periodicity and data without it are mixed together in View redraw behaviors.

2) Result without Advertisement: Fig. 5 shows the invocation time of HTTP connection and View redraw behaviors when the application C without advertisement was run on Android emulator. In this figure, HTTP connections and View redraw behaviors are invoked many times. However, at the whole period of the log data, these two behaviors are not invoked at a specific interval. In other words, two behaviors don't have periodicity.

D. The feature of the target data

In this paper, the targets of analysis are two sequential data which correspond to the invocation time of HTTP connections and View redraw. As mentioned in Section IV-A, when an advertisement is contained in an application, periodicity appears in these two behaviors. However, in Android systems, since the operations executed by application code and by advertisement code are mixed up, non-periodic data may be added into the periodic data. Fig. 4 is an actual example for that case. Therefore, we need to examine the analysis technique which the following requirements are satisfied with.

1) The periodicities in two sequential data can be compared with.
2) Data can be analyzed even if the non-periodic data are mixed into it.

V. Detection Method

As mentioned in Section IV-A, if an application has advertisement, the invocation time of HTTP connections and View redraws has a certain periodicity. In this section, we discuss how it is appropriate to analyze two data (the invocation time of HTTP connections and View redraws) to judge the existence of an advertisement in an application.
As the analysis method which satisfies requirements mentioned in Section IV-D, we propose the analysis in the frequency domain using Fourier transform. By Fourier-transforming the sequential data and analyzing these data in frequency domain, the noise-proof can be increased and the influence of execution lags of an application will be reduced.
Table I shows steps of the analysis in the proposal method. In the following sub sections, we will explain about each step.

TABLE I. Steps of the analysis in proposal method

1. Sampling log data for Fourier transform
2. Execution of Fourier transform
3. Calculation of correlation coefficient
4. Evaluation using Logistic regression

A. Sampling log data

Firstly, we sample an application's log data for Fourier transform. Fig. 6 is an example of sampled data. This data starts from the time of an application is launched. The first row expresses the time. In the sampled data, the time is divided into intervals of a certain as a time slot. Fig. 6 is the data in the case of setting width of time slot to 5 seconds. The second row corresponds to the existence of HTTP connection. In the time slot of each line, when an application establishes HTTP connection, value "1" is output. And, when an application doesn't establish HTTP connection, value "0" is output. The third row corresponds to redraw of View behavior. Like the second row, in each time slot, when an application performs redraw of View, value "1" is output and when an application doesn't, value "0" is output. In the following sections, these two sequential data are Fourier-transformed and we analyzed these data in frequency domain.

B. Fourier Transform

After the log data is sampled for Fourier-transform, the sampling data is Fourier-transformed. Therefore, the data analyzed in the following steps are the waveforms whose the

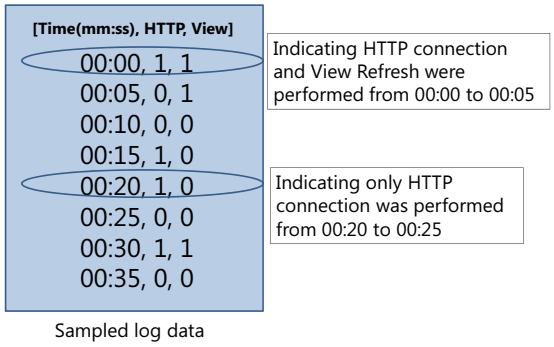

Fig. 6. An example of sampling data for Fourier transform when setting the time slot 5 sec

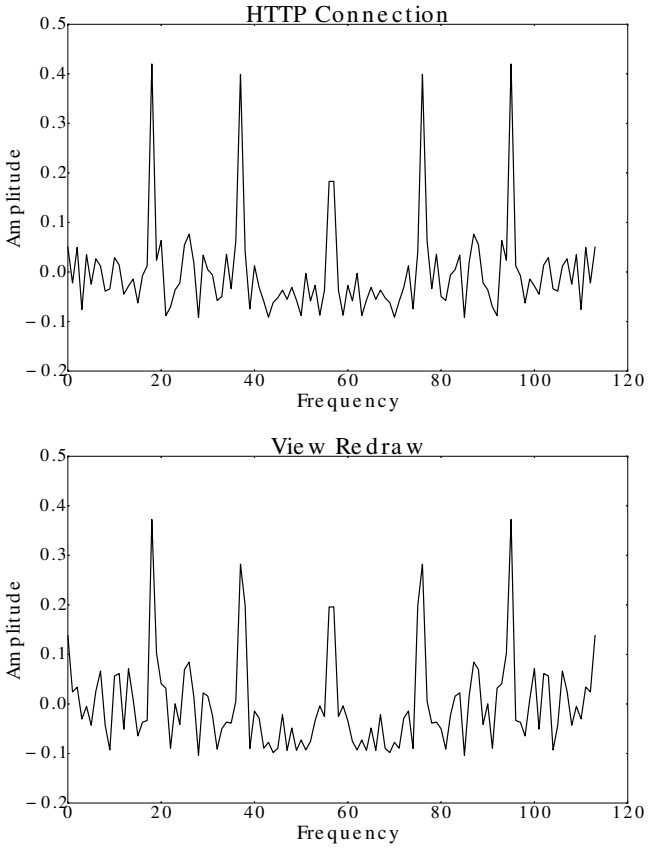

Fig. 7. An example of the wave form of Fourier-transformed data produced from an application with advertisement

horizontal axis indicates the frequency and the vertical axis indicates the amplitude. Fig. 7 is the waveform produced by executing Fourier-transform for the log data of Application B shown in Fig. 4. In this figure, the horizontal axis indicate the label of frequency using serial numbers.

C. Detection method using correlation coefficient

We performed detection method using data in the frequency domain obtained in Section V-B. In this paper, we consider that detection of advertisement can be obtained by using the correlation coefficient of these two data. Equation 1 shows the definition of correlation coefficient. We adapted two output waveforms shown in Fig. 7 to the x and y in 1 respectively. As mentioned above, when an advertisement is contained in an application, periodicity appears in two data. In the Fourier-transformed data, this periodicity appears as a specific frequency component. In Fig. 7, five peak values are equivalent to this frequency component. When calculating correlation coefficient, average value is subtracted from each x and y as shown in 1. So, only five peak values have major influence on the output value of a correlation coefficient in our method because the other frequency components are settled near 0. Therefore, when the log of an application that has advertisement is analyzed, the correlation coefficient of two data should become high value close to 1.
After outputting the values of correlation coefficient for each application's log data, we examined detection rate of proposal method using logistic regression.

$$\frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}} \tag{1}$$

VI. Experiment

A. Experimental procedure

In this paper, we prepared 24 log data of applications with advertisement and 21 log data of applications without advertisement for experiment. We acquired these log data by running sample applications on Android emulator which is based on modified OS to record HTTP connection and View redraw behavior. The sample applications are a part of applications which ran normally on Android emulator from free applications exhibited in Google Play. For example, game applications which use much GPU power or map applications using Google Map API cannot run on Android emulator normally. So, we cannot use such applications' log data.
In the experiment, we evaluated accuracy rate and false positive of detection advertisement using the method mentioned in Section V. In order to reduce the deviation arising from the smallness of log data, we used cross-validation. Dividing advertisement sample group and non-advertisement sample group into three groups severally, we used one group for learning phase and two group for validation phase to output detection rate. Then, the combination of learning data and validation data was rotated, and validation was executed in three kinds of combinations. In our method, the output value has randomness since Stochastic Gradient Descent method is used in logistic regression. So, we repeated this calculation a hundred times, and we defined an average value as a result for detection rate and false positive.
In this experiment, we changed the width of time slot as a parameter. We performed experiment with four parameters, namely, 1 sec, 2 sec, 5 sec and 10 sec. Also, in this paper we cut first 1 minute data from each log data. This is because advertisement library perform initial operations at the time of launch and this influence on the periodicity of log data. After

TABLE II. DETECTION RATE AND FALSE POSITIVE FOR FOUR PARAMETERS

Width of time slot	accuracy(%)	false positive(%)
1s	75.53	30.64
2s	76.20	29.79
5s	75.57	32.33
10s	72.64	33.50

cutting initial operations, we used 10 minutes log data for the analyze. Table II shows the result of the experiment.

B. Consideration

In this paper, we performed the experiments for four width value of time slot as parameters in sampling of log data. The parameters are 1 sec, 2 sec, 5 sec and 10 sec. From the outputted results, we can consider following things.
First, even if the parameters are changed, it was not affected largely to a detection rate. It seems that many advertising libraries performs updating of advertisement image at intervals for 30 seconds to 1 minute. Therefore, it seems that small change of time slot width didn't majorly influence the detection rate. Also, end-to-end delays of advertising image fetching does not majorly influence the analysis because such delays are smaller than the period of ad behaviors.
We also considered a cause of false positive. One of the causes is that some unexpected frequency components may appear in log data . Even if an advertisement library was not incorporated into an application, there is a possibility that any frequency component appears between the HTTP and View behavior when these two behavior are invoked many times. Also, as another case, if an application's code invokes HTTP and View behavior which have the periodicity, this may detected as the operation of an advertising library. We will discuss how to solve these problems in the next section.

VII. FUTURE WORKS

All of HTTP connections and View object redraw behaviors are recorded on log data. Therefore, as shown in Fig.4 and Fig.5, behaviors related to advertisement and not related to advertisement are mixed together in log data. There is a possibility that this causes false positive.
By grouping HTTP connections and View behaviors into some series based on related information and analyzing log data for each series, the above problem may be solvable. For example, there is a method focusing on destination IP address in HTTP connections. Consider the case of running a news application which has an advertisement library. In this case, this application performs HTTP connections for acquisition of advertising images and news information. Generally, advertisement images are provided from a server managed by advertisement providers. On the other hand, news information is provided from another server. So, HTTP connections performed by the application have two destinations, a server providing news and a server providing advertisement. Therefore, destination IP addresses of HTTP connections which appeared in that news application's log data can be grouped into two series. Thus, an advertisement and the original function of an application can be divided by grouping HTTP connections. This may make accuracy rate to rise.

VIII. CONCLUSION

In this paper, a detection method of advertisement library focusing on HTTP connections and View object redraw behaviors is proposed. Mobile advertisement libraries update advertising image periodically to attract a user's attention. This means that the invocation times of HTTP connections and View object redraw have a certain periodicity by an application with advertisement. We realized detection method of advertisement library by extracting this periodicity using a combination of Fourier-transform and correlation coefficient. With this method, it is possible to detect advertisement library only from its behavior. Because we focus on the behavior of advertisements in our method, it is unnecessary to acquire signatures of advertisement libraries in advance. Therefore, unknown advertisement library can be detected with proposal method. And, since our method is based on the View object which is the component of applications' screen, our method is hard to be affected to the small difference in implementing for every advertising library. Our proposal method resulted in a output of about 76 % detection rate for about 50 sample applications.

ACKNOWLEDGMENT

We would like to thank Ayumu Kubota and Takamasa Isohara, KDDI R&D Labs for giving beneficial advices in early work of this research.

REFERENCES

[1] "Android operating system statistics - appbrain." [Online]. Available: http://www.appbrain.com/stats/stats-index

[2] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo, "Don't kill my ads!: balancing privacy in an ad-supported mobile application market," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 2.

[3] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Investigating user privacy in android ad libraries," in *Workshop on Mobile Security Technologies (MoST)*. Citeseer, 2012.

[4] H. Kuzuno and K. Magata, "Detecting advertisement module network behavior with graph modeling," in *Proceedings of the 9th Asia Joint Conference on Information Security(AsiaJCIS2014)*. IEEE, 2014.

[5] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner, "Addroid: Privilege separation for applications and advertisers in android," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 71–72.

[6] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, and W. Zou, "Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications," in *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2012, pp. 93–104.

[7] Y. Nishimoto, N. Kajiwara, S. Matsumoto, Y. Hori, and K. Sakurai, "Detection of android api call using logging mechanism within android framework," in *Security and Privacy in Communication Networks*. Springer, 2013, pp. 393–404.

[8] "Connecting to the network — android developers." [Online]. Available: http://developer.android.com/training/basics/network-ops/connecting.html