Vol.2017-CSEC-78 No.31
Vol.2017-SPT-24 No.31
2017/7/15

Androidアプリケーションにおける Runtime Permission要求時の挙動調査

河合 佑紀[1]　明田 修平[1]　半井 明大[2]　窪田 歩[2]　瀧本 栄二[1]　毛利 公一[1,a)]

概要：スマートフォンはユーザが好みのアプリケーションをインストールできるため人気が高い．しかし，ユーザに無断で個人情報にアクセスし漏洩させるアプリケーションの存在が指摘されている．例えばAndroidでは，アプリケーションによるデータへのアクセスはAndroid Permissionと呼ばれる権限によって制御されている．しかし，Android 5.1以前では，パーミッションの要求がインストール前の一度しか行われず，ユーザがその時点でアプリケーションが信用できるものか判断する必要がある．そのため，ユーザの中には，アプリケーションを利用したいがためにAndroid Permissionを理解せずにインストールを許可してしまうという課題がある．これに対して，Android 6.0以降ではRuntime Permissionと呼ばれる機能が追加された．本論文では，Runtime Permissionによって課題が改善されているかを確認するため，実際のアプリケーションマーケットからアプリケーションを収集し，どのようにRuntime Permissionを用いてAndroid Permissionが要求されているかを調査した．

Behavior Examination of Android Applications Requesting Runtime Permissions

Yuki Kawai[1]　Shuhei Aketa[1]　Akihiro Nakarai[2]　Ayumu Kubota[2]　Eiji Takimoto[1]
Koichi Mouri[1,a)]

Abstract: Smartphones are now very popular because users can install applications according to their preference. However, existence of applications that access personal information and leak them is pointed out. Generally, in Android, accesses by applications to data is controlled by Android Permission mechanism. In Android 5.1 or earlier, it requests permissions to a user only once before installation of an application. The user must decide whether the application is trusted or not at that time. Therefore many users give permission without enough understanding because they want to use the application soon. On the other hand, in Android 6.0 or later, the mechanism called Runtime Permission was added. It is supposed to solve the problems of older versions of Android. So we collected applications from an existing and famous application market. Then we examined how they request permissions to user using Runtime Permission mechanism.

1. はじめに

スマートフォンは，ユーザがマーケットからアプリケーション (以下，Appと記す) をインストール可能であり利便性が高い．しかし，スマートフォンは個人的な情報が格納されている場合が多いため，それらの情報の扱い，特にAppからのアクセスの扱いは慎重になされなければならない．本論文では，Google社のAndroidを例にその扱いについて調査を行ったので，それについて報告する．

Androidでは，Appによる利用者情報へのアクセスをAndroid Permission(以下，パーミッションと記す) と呼ばれる権限を用いて制御している．ユーザがパーミッションを許可すると，Appは利用者情報を取得することができる．パーミッションの要求方法は，Android 5.1以前とAndroid 6.0以降で異なる．5.1以前では，Appがインス

[1] 立命館大学
Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525–8577, Japan
[2] 株式会社KDDI総合研究所
KDDI Research, Inc.
[a)] mouri@asl.cs.ritsumei.ac.jp

トールされる直前にユーザに対してパーミッションの要求が行われる．また，一度許可されたパーミッションをユーザが不許可に変更することはできない．このように 5.1 以前では，パーミッションの要求がインストール直前にしか行われないため，ユーザは App のインストール前に App が信用できるか否かを判断する必要がある．しかし，スマートフォンプライバシーイニシアティブ [1]（以下，SPI）の調査によると，ユーザのプライバシ保護に対する意識は低く，49.4%の Android のユーザは App のインストール時に求められるパーミッションを理解せずに同意し，App を使用している．その結果，ユーザの意図しないところで App による利用者情報の取得・流出が起こるという課題が発生している．

この課題に対して，Android 6.0 以降では，Runtime Permission が追加された．Runtime Permission は，実行時にユーザに対してパーミッションの要求を行う機構である．また，インストール後であっても，ユーザは App 毎にパーミッションの許可・不許可を変更できる．そのため，App のインストール前にパーミッションが要求されることはない．Runtime Permission を利用するメリットとして以下の 3 点を挙げることができる．

- App は，パーミッションが必要になるタイミングでユーザに対して許可を求めることができる．
- App は，パーミッションの要求前にその利用目的についてユーザに説明することができる．
- ユーザは，Setting App と呼ばれる設定管理アプリケーションを用いてパーミッションの許可・不許可をいつでも変更できる．

ただし，どのモジュールがどのタイミングでパーミッション要求を行うかについては，App 開発者に委ねられている．Google が提供する Runtime Permission に対するガイドでは，App 開発者に対し，App がパーミッションを要求する直前にパーミッションの利用目的等の説明を行うことを推奨している [2]．これは，説明によってユーザが，要求されたパーミッションを理解できるようにするためである．しかし，次のような形でパーミッションの要求がなされる場合が考えられ，Runtime Permission を用いた場合であっても，パーミッションに対するユーザ意識の課題が Android 5.1 以前の場合と同様に発生してしまう可能性がある．

- App の起動直後に複数のパーミッションを一括して要求する．
- App によるパーミッションの要求に対して，説明がなされていない．
- 外部モジュールによって，App 本来の目的とは関係のないパーミッションの要求がなされる．

以上の背景から，本論文では，Android 6.0 におけるパーミッションの要求に関する課題の有無を調べるべく，Runtime Permission の利用に関する実態を把握することとした．具体的には，Google Play[3] で配布されている App を実際に動作させ，パーミッション要求時の画面と要求元モジュールを基に分類し，App がどの程度 Google のガイドに沿っているかを調査した．

表 1 利用者情報とその取得に必要なパーミッション

利用者情報の種類	内容	取得に必要なパーミッション
利用者の識別に係わる情報	氏名，住所等の契約者情報	GET_ACCOUNTS READ_PHONE_STATE
	アカウント認証に必要な識別情報	GET_ACCOUNTS
	Cookie 技術を用いて生成された識別情報	パーミッションは不要
	契約者・端末固有 ID	READ_PHONE_STATE ACCESS_NETWORK_STATE ACCESS_WIFI_STATE
第三者の情報	連絡帳で管理されるデータ	READ_CONTACTS
通信サービス上の行動履歴や利用者の状態に係わる情報	通信履歴	READ_CALL_LOG READ_SMS
	Web ページ上の行動履歴	com.android.browser.permission. READ_HISTORY_BOOKMARKS
	App の利用履歴	NFC READ_LOG
	位置情報	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
	写真，動画	CAMERA

表 2 Dangerous Level のパーミッションと Permission Group

Permission Group	Permission
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARCE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALL
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

2. Android Permission の概要

App 開発者は，App で必要となるパーミッションを，AndroidManifest.xml ファイルに記述しなければならない．AndroidManifest.xml は，App の名前や Activity などの App に関する情報が記述されているファイルである．パーミッションを必要とする API のうち，利用者の識別，第三者に関する情報，行動履歴や利用者の状態など，利用者情報に関連するものに**表 1** のようなものがある．

パーミッションには，アクセスする情報に応じてプロテクションレベルが設定されている．Android で定義されているプロテクションレベルは，以下の 4 種類である．

Dangerous Level

利用者情報へアクセスする可能性が高いパーミッション群に対して定義される．App がこのレベルのパーミッションを必要とする場合ユーザに対して明示的に許可を得る必要がある．具体的なパーミッションおよびそれが属する Permission Group を**表 2** に示す．

Normal Level
このレベルのパーミッションを必要とする場合でもユーザに対して明示的に許可を求める必要はない．App のインストール時に自動で権限が付与される．

Signature Level
パーミッションを定義している別の App と署名が一致した場合にのみ自動で権限が付与される．

SignatureOrSystem Level
Android の端末にプリインストールされている App と署名が一致した場合にのみ自動で権限が付与される．

3. マーケット App の調査に必要な情報を取得する方法

App 開発者は，android.app.Activity.requestPermissions API を利用してパーミッションを要求するためのダイアログを出力させる．ダイアログ表示のタイミングや説明の有無を App 開発者が自由に決定できるため，パーミッション要求の挙動は App 毎に異なる．App を調査するために，以下の 3 つの情報を取得する．

- AndroidManifest.xml で宣言されているパーミッション: ダイアログを出力する App の判定，どのような利用者情報を取得する App なのかを把握する．
- android.app.Activity.requestPermissions API の呼出し元モジュール: どのモジュールがパーミッションを要求しているかを把握する．
- ダイアログが表示される前後の画面: パーミッション要求に対して，説明の有無や要求のタイミングを確認する．

これらの調査結果を用いて，App のどのモジュールから要求されているか，ユーザがパーミッションについて理解できる要求方法であるかなどを判断し，パーミッション要求に関する挙動を把握する．以下，それぞれについて述べる．

3.1 AndroidManifest.xml ファイルで宣言されているパーミッションの取得手法

Runtime Permission の対象となるパーミッションは，Dangerous Level で定義されるパーミッションである．また，App が，Dangerous Level のパーミッションを要求するためには AndroidManifest.xml ファイルに記述する必要がある．そのため，AndroidManifest.xml ファイルに記述されている Dangerous Level のパーミッションを観測することで，利用者情報の利用を把握することができる．以下に，調査方法の手順を示す．

(1) Android SDK のツール aapt を用いて apk ファイル内の AndroidManifest.xml に記述されているパーミッションを取得する．
(2) 取得したパーミッションの一覧から Dangerous Level のパーミッションを抽出する．

3.2 android.app.Activity.requestPermissions API の呼出し元の取得手法

Android では，API 呼出しまでの一連の呼出し過程をスタックトレース情報から取得することで，API を呼び出したモジュールを特定できる．本論文では，App のソースコードにスタックトレース情報を出力するためのコード挿入することで呼出し元 API を取得する手法 [4] を用いる．この手法は，次のような手順で呼出元の特定を実現している．

(1) apk ファイルを apktool[5] を用いてデコードし，バイトコードの逆アセンブル結果である smali ファイルを取得する．
(2) smali ファイルに対してスタックトレース情報を出力するコードを挿入する．
(3) コード挿入した smali ファイルを apktool を用いて再コンパイルし apk ファイルを作成する．
(4) 作成した apk ファイルに署名する．
(5) 書き換えた App を端末にインストールし実行することで android.app.Activity.requestPermissions() の呼出し元モジュールを観測する．

3.3 動画を用いた App 動作中の画面取得手法

App 調査において，パーミッション要求に対する説明の有無や要求のタイミングを確認するために，App の動作中の画面を取得する．具体的には，動画を用いた App 動作中の画面を取得する手法を用いる．App の動作を動画として取得することで，App の実装に関わらず調査に必要な画面を抽出することができる．動画の記録とダウンロードには adb (Android Debug Bridge)[6] コマンドを用いた．また，動画からダイアログ出力前後の画面の画像を抽出するのには MoviePy[7] を用いた．MoviePy では，コマンドラインで指定した動画ファイルについて，指定した時刻の画像を抽出する機能を有する．具体的な手順を次に示す（**図 1** 参照）．

(1) $ adb shell screenrecord コマンドを用いて動画の記録を開始する（時刻 t_1）．
(2) 調査対象となる App を Android で実行させる．
(3) android.app.Activity.requestPermissions API によるダイアログの出力を確認する（時刻 t_2）．
(4) 録画を停止させ動画ファイルを生成させる．
(5) $ adb pull コマンドで，Android 内に生成された動画

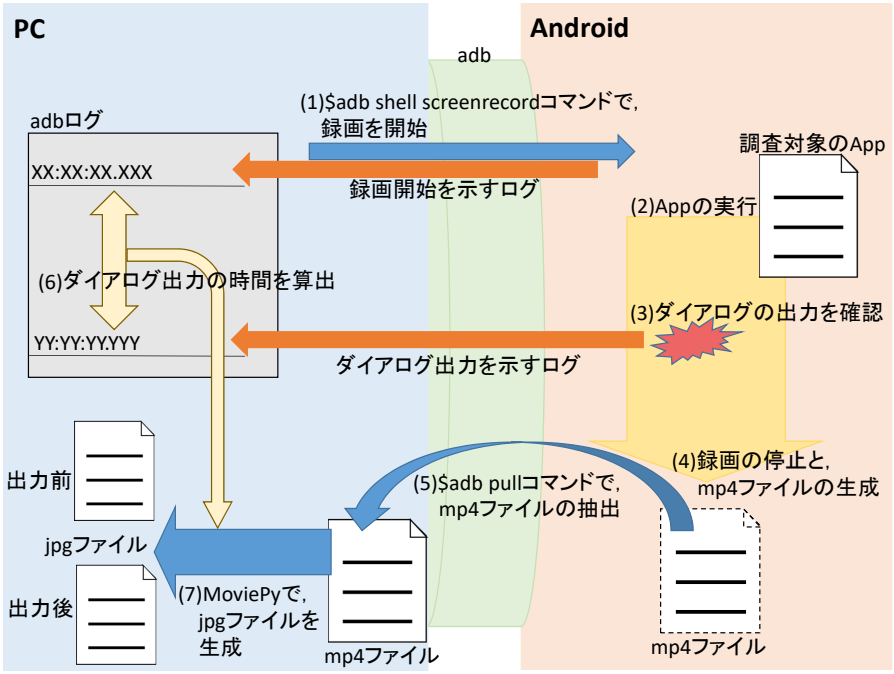

図 1　動画の取得から画像の抽出までの一連の手順

ファイルをダウンロードする．

(6) 時刻 $t_2 - t_1$ から動画内におけるダイアログ出力の時間を特定する．

(7) (6) で特定した時間を基に MoviePy を用いてダイアログ出力前後の画像をファイルに保存する．

4. マーケット App の調査

4.1 調査環境

Runtime Permission によって Android 5.1 以前にあったパーミッション要求の課題が改善されているかを明らかにするため，App によってどのようにパーミッションが要求されているか調査を実施した．調査に利用する App は，Runtime Permission で動作する可能性の高いものが望ましい．そこで，次のような条件で調査した．

- 利用する App は，Google Play の新着無料 App 上位 50 個を取得した．
- 2016 年 10 月 5 日から 2016 年 12 月 14 日の期間について 2 週間ごとに調査日を設けた．
- 調査環境は Android 6.0 を搭載した Nexus6P の実機を用いた．

App の動作を把握するために，3 章で述べた手法を用いて，要求するパーミッションの種類，ダイアログ出力前後の画面，android.app.Activity.requestPermissions API の呼出し元モジュールを確認する．それらの結果から調査対象の App を以下に示す内容で分類する．

- 要求のタイミング
- 説明の有無
- 許可前後の挙動でパーミッションを利用した利用者情報へのアクセスが分かるかどうか
- パーミッションを要求するダイアログの呼出し元モジュール

4.2 調査結果

表 3 は，調査期間中の各調査日において，App の挙動を，ダイアログ出力のタイミングやパーミッションに対する説明の有無で分類した結果を示している．例えば，10 月 5 日に取得した 50 個の App のうち，Runtime Permission で動作する条件を満たした App は 29 個であった．その 29 個の App を動作させたところ，24 個の App でダイアログの出力が確認できた．また，ダイアログが確認できた 24 個の App のうち，21 個の App で起動直後にダイアログが表示され，ダイアログ出力の前後の画面からパーミッションの利用が分かる App は 2 つであった．パーミッションに対する説明ついては，7 個の App で説明があり，17 個の App で説明がなかった．ダイアログの呼出し元が外部モジュールである App は 16 個存在した．

今回の調査では，調査日をまたいで重複する App が存在した．そこで，Runtime Permission を利用できるかつ Dangerous Level のパーミッションを要求する App に対して，重複する App を除き，全調査日の結果を集計した．**表 4** は，集計結果を示す．この結果より，各項目の動作を行う App の割合が分かる．

4.3 考察

Runtime Permission を利用することで，Android 5.1 以前の課題が改善されていると考えられる動作は 2 つある．

表 3　調査日ごとの調査結果

	10/5	10/19	11/2	11/16	11/30	12/14
取得した App	50	50	50	50	50	50
RP と DLP の両方を使用	29	31	25	26	21	15
ダイアログを表示した	24	20	17	11	10	6
起動直後にパーミッションを要求	21	20	15	9	10	6
ダイアログ表示前後の挙動で利用が分かる	2	0	1	1	1	0
パーミッションの説明あり	7	5	6	3	2	2
パーミッションの説明なし	17	15	11	8	8	4
ダイアログ呼出し元が外部モジュール	16	16	11	8	7	5

RP: Runtime Permission，DLP: Dangerous Level のパーミッション

表 4　調査期間に取得した App の集計結果

RP と DLP の両方を使用	112
ダイアログを表示した	64
起動直後にパーミッションを要求	58
ダイアログ表示前後の挙動で利用が分かる	5
パーミッションの説明あり App	18
パーミッションの説明なし App	46
ダイアログ呼出し元が外部モジュール	46

1 つは，App がパーミッションの用途が分かるような動作をダイアログ表示の前後で行っている場合である．そのような動作をする App の例を**図 2** に示す．

図 2(1) の画面で，カメラ起動と書かれたボタンをタップするとパーミッション要求のダイアログが出力される．ユーザが，そのダイアログに対して許可をタップすると図 2(3) の画面にあるようにカメラ機能が動作する．このように，ユーザに対してカメラを利用することが分かるようなパーミッション要求方法は，Android 5.1 以前の課題が改善されている要求方法であると考えられる．今回の調査において，パーミッション要求の前後の画面からパーミッションの用途が分かる App は 5 個であり，ダイアログが確認できた App における割合は 7.8%と少ない．割合が少なくなった要因として，App が要求するパーミッションの種類が考えられる．PHONE グループや CONTACTS グループのパーミッションなどは，端末固有の ID や連絡帳などのデータの取得に利用されるため，App がパーミッションを利用してデータを取得したことが画面上に現れにくい．その結果，今回の調査では App の画面からパーミッションの用途が分かる App が少なかったと考える．一方で，パーミッション要求の前後の画面からパーミッションの用途が分かる App は，CAMERA グループや STORAGE グループのパーミッションを要求していた．これらのパーミッションで App がアクセスできる情報は，カメラ機能におけるデータや SD カード内のデータといった App の画面上に表示しやすいものである．

もう 1 つは，パーミッションを要求するダイアログが出力される前後で，パーミッションに対する説明がなされている場合である．そのような動作をする App の例を**図 3** に示す．

図 3 の App は，(1) の画面で STORAGE グループのパーミッションを要求するダイアログを出力している．その後に表示される図 3(2) の画面では，ダイアログの表示の後に要求した STORAGE グループのパーミッションをどのような目的で利用するか説明文が表示されている．このように，ユーザに対してパーミッションの利用目的が明示されている App も，Android 5.1 以前の課題が改善されていると考えられる．

しかし，パーミッションに対する説明が無い App は 46 個とダイアログが確認できた App における割合は 71.8%である．また，App の起動直後にパーミッションを要求している App が 58 個とダイアログが確認できた App における割合は 90.6%である．調査結果では，パーミッションに対するユーザ意識の問題が，Android 5.1 以前のパーミッションの要求手法と同様に発生してしまう可能性のある App が多く見られた．その原因として，ゲームエンジンによる要求が考えられる．ゲームエンジンは，ゲーム App の開発に利用できるひな形やライブラリを提供しており，App 開発者の支援を目的にした外部モジュールである．今回の調査期間における新着無料の App にはゲーム App が多く，それらの App の中には，ゲームエンジンモジュールを組み込んでいる App が多く存在した．当該モジュールによるパーミッション要求の場合，App はパーミッションに対する説明を出さず，App の起動直後にパーミッションを要求している．App 内部のモジュールからパーミッションを要求している場合は説明のあることが多いため，このような外部モジュールによるパーミッション要求は，App 開発者も意図していない可能性が考えられる．App 開発者は，パーミッションにおける課題を改善するために，自らが組み込んだ外部モジュールが要求するパーミッションに関しても説明する必要がある．

5. 関連研究

Andriotis ら [8] は，Settings App に注目し，Android 6.0 のユーザがどの程度 Runtime Permission の機能に適応しているか調査を行っている．Andriotis らは，調査対象の

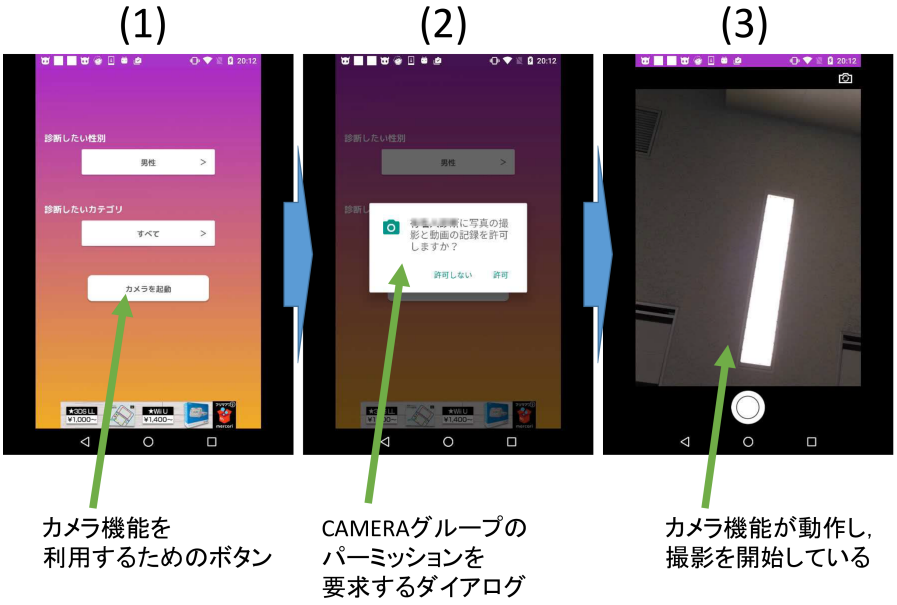

図 2　パーミッションの用途が分かるような動作をダイアログ表示の前後で行っている App

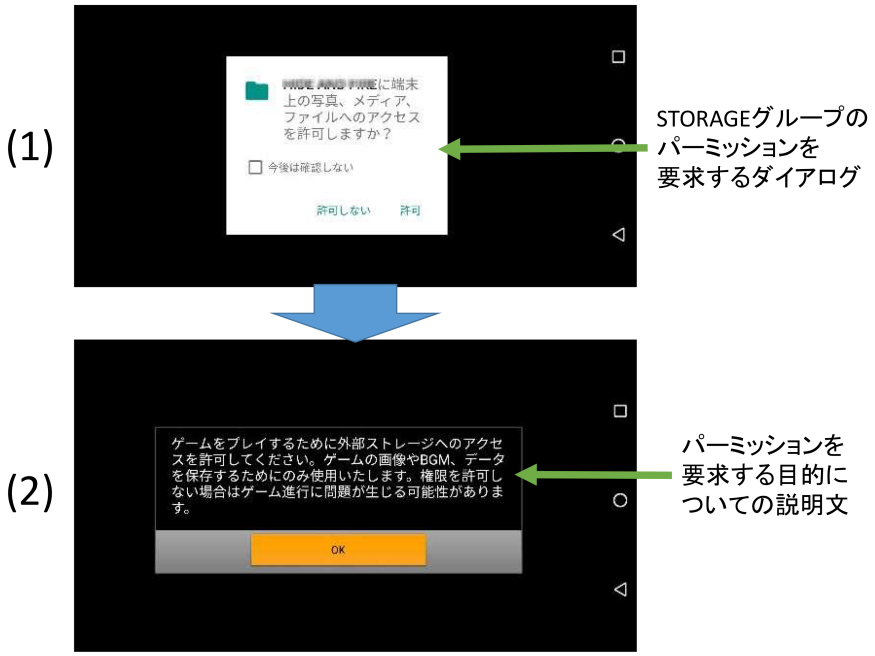

図 3　パーミッションに対する説明がなされている App

ユーザがインストールしている App の Package 名とその App が要求しているパーミッションを収集し，パーミッショングループごとに要求される割合を出している．調査結果として，Andriotis らは，約 60%のユーザが Runtime Permission の機能に適応し，ユーザ自らパーミッションを設定できているとしている．しかし，Andriotis らの研究では，App の実行中におけるパーミッション要求に関しての言及は行われていない．そのため，Android 5.1 以前の課題に対して，Runtime Permission を利用したパーミッション要求が，どの程度対応しているかは明らかになっていない．

また，利用者情報の取得をユーザーに通知する手段として App マーケットにおける説明文に着目し，調査した研究がある [9]．渡邉らは，独自のフレームワークを開発し，App の挙動と App マーケットにおける説明文の相関性を明らかにしている．渡邉らは，調査対象の App マーケットを Google Play だけでなく，サードパーティマーケットで公開されている App についても調査している．この調

査において，渡邉らは，利用者情報を連絡帳にある情報，位置情報，カメラ機能の 3 つに限定し，計 20 万個という膨大な数の App について調査している．この調査結果では，利用者情報の取得について説明文で言及している App は 4–33%であるとしている．さらに，渡邊らは，パーミッションのフィルタリングやメソッドのコールグラフによって App の挙動をコード解析している．その解析の結果から渡邊らは，App 開発者が利用者情報について説明文で言及しない理由のいくつかを明らかにしている．その理由のひとつとして，広告目的の外部モジュールによる利用者情報へのアクセスが挙げられている．本論文においても，パーミッションに対して説明のなされていない App には，外部モジュールがパーミッションを要求している App が多数存在した．そのため，外部モジュールによる利用者情報へのアクセスに対して，対策が必要であると考える．

6. おわりに

本論文では，Runtime Permission が動作する App に対する，パーミッション要求時の動作の調査に必要な情報を取得するための手法を提案し実装したうえで，App の実態調査を行った．

パーミッション要求時の動作調査手法では，調査対象の App に対してコード挿入することでスタックトレース情報を出力させ，パーミッションの要求元モジュールを特定する手法を実現した．また，adb コマンドを利用することで App の動作画面を動画として取得し，App がパーミッションを要求する前後の画面を取得する機能も実現した．

App 調査では，Runtime Permission を利用したことで Android 5.1 以前の課題が改善されていると考えられる動作をする App を確認できた．しかし，調査対象となった App の 70-90%が，パーミッションを起動直後に要求する App やパーミッションに対する説明をしない App であった．また，外部モジュールを含んだゲーム App にこのような動作が多く見られたため，外部モジュールを利用するゲーム App の開発者に対して，パーミッションの課題を改善するよう対策を講じる必要がある．この結果から，Google Play で公開されている App 全体として，Runtime Permission によるパーミッションの課題のさらなる改善が必要であると考えられる．

参考文献

[1] 総務省：スマートフォン プライバシー イニシアティブ 利用者情報の適正な取扱いとリテラシー向上による新時代イノベーション，利用者視点を踏まえた ICT サービスに係る諸問題に関する研究会 (2012).
[2] Google: Requesting Permissions at Run Time, https://developer.android.com/training/permissions/requesting.html (2017 年 1 月 30 日閲覧).
[3] Google: Google Play, https://play.google.com/store/apps?hl=ja (2017 年 1 月 30 日閲覧).
[4] 岩田直樹，明田修平，瀧本栄二，川端秀明，半井明大，窪田 歩，毛利公一：Android アプリケーションへのコード挿入を用いた API 呼出し元モジュール特定手法，2017 年暗号と情報セキュリティシンポジウム (SCIS2017) (2017).
[5] Apktool: iBotPeaches, https://ibotpeaches.github.io/Apktool/ (2017 年 1 月 30 日閲覧).
[6] Google: Android Debug Bridge, https://developer.android.com/studio/command-line/adb.html (2017 年 1 月 30 日閲覧).
[7] Zulko: MoviePy, http://zulko.github.io/moviepy/# (2017 年 1 月 30 日閲覧).
[8] Andriotis, P., Sasse, M. A. and Stringhini, G.: Permissions Snapshots: Assessing Users' Adaptation to the Android Runtime Permission Model, *Proc. of Workshop on Information Forensics and Security (WIFS) 2017*, IEEE, pp. 134–146 (2016).
[9] 渡邉卓弥，秋山満昭，酒井哲也，鷲崎弘宜，森 達哉：Android アプリの説明文とプライバシー情報アクセスの相関分析，コンピュータセキュリティシンポジウム 2014 論文集，Vol. 2014, No. 2, pp. 590–597 (2014).