

推薦論文

# 静的解析による Android パーミッションの利用目的の可視化方法

坂下 卓弥<sup>1,a)</sup> 小形 真平<sup>1,b)</sup> 海谷 治彦<sup>1,†1,c)</sup> 海尻 賢二<sup>1,d)</sup>

受付日 2014年3月13日, 採録日 2014年10月8日

**概要:** Android には利用者情報漏えい等を防ぐためのパーミッションシステムがある。このシステムは、アプリケーションが「連絡先データの読み取り」や「完全なインターネットアクセス」等のパーミッションを利用することの許可をインストール時にユーザに求める。しかし、マルウェアによる利用者情報漏えいが生じていることから、当該システムは十分に機能していない。我々はこの主原因を当該システムがパーミッションの利用目的をユーザに通知しない点にあると仮定した。利用目的とは、アプリケーションが「端末の電話番号」や「電話帳のメールアドレス」等の情報をどこの URL 等へ送信しているかを指す。そこで、本研究ではユーザがパーミッションの利用目的を理解しやすいように、アプリケーションを静的解析し、利用目的を可視化する方法を提案する。本論文では、提案手法はパーミッションシステムよりも既存のマルウェアを判別容易にするかを評価する。

**キーワード:** Android, スマートフォン, 静的解析, 利用者情報, パーミッション, 可視化

## Static Analysis for Highlighting what Android Application Does with Permissions

TAKUYA SAKASHITA<sup>1,a)</sup> SHINPEI OGATA<sup>1,b)</sup>  
HARUHIKO KAIYA<sup>1,†1,c)</sup> KENJI KAIJIRI<sup>1,d)</sup>

Received: March 13, 2014, Accepted: October 8, 2014

**Abstract:** Android has a permission system to prevent identity theft and so on. The system makes an android user grant the permissions such as “Full internet access” and/or “Read your contacts” to an application. However, this system is not enough to prevent identity theft because a lot of the identity theft has occurred. We assumed that the main problem of the theft is caused by the system which does not notify the user the purpose of the application. Here, the purpose implies that the sensitive data such as “Phone number,” “e-mail address,” etc. are sent to a URL. We therefore propose a static analysis method for highlighting what the application does with permission so that the user can understand the purpose of the application. In this paper, we evaluate our method whether it can provide us more sufficient information for determining malware correctly than the existing permission system.

**Keywords:** Android, smartphone, static analysis, user information, permission, highlight

### 1. はじめに

近年、スマートフォンの代表的な OS である Android の普及が急速に進んでいる。Android では、Android アプリ

本論文の内容は 2013 年 7 月の CSEC62 研究発表会にて報告され、同研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である。

<sup>1</sup> 信州大学  
Shinshu University, Nagano 380–8553, Japan  
<sup>†1</sup> 現在、神奈川大学  
Presently with Kanagawa University  
<sup>a)</sup> 13tm518g@shinshu-u.ac.jp  
<sup>b)</sup> ogata@cs.shinshu-u.ac.jp  
<sup>c)</sup> kaiya@kanagawa-u.ac.jp  
<sup>d)</sup> kaijiri@cs.shinshu-u.ac.jp

表 1 利用者情報の種類\*1  
Table 1 Types of user information.

| 区分                        | 情報の種類          | 含まれる情報                                                       |
|---------------------------|----------------|--------------------------------------------------------------|
| 第三者の情報                    | 電話帳で管理されるデータ   | 氏名, 電話番号, メールアドレス等                                           |
| 利用者の識別に係る情報               | 氏名, 住所等の契約者情報  | 電話番号                                                         |
|                           | 契約者・端末固有 ID    | IMSI, ICCID, IMEI, MAC アドレス等                                 |
| 通信サービス上の行動履歴や利用者の状態に関する情報 | 通話履歴           | 通話内容・履歴, メール内容・送受信履歴                                         |
|                           | ウェブページ上の行動履歴   | 利用者のウェブページ上における閲覧履歴                                          |
|                           | アプリケーションの利用履歴等 | アプリケーションの利用履歴・記録されたデータ等, システムの利用履歴等                          |
|                           | 位置情報<br>写真・動画等 | GPS 機器によって計測される位置情報, 基地局に送信される位置登録情報<br>スマートフォン等で撮影された写真, 動画 |

ケーションの配信サービスである Google Play を通じて Android アプリケーションを誰でも自由に公開できるため, 年間数十万のアプリケーションが公開されている。

しかし, Android の普及にともない, 利用者情報 [1] を外部に流出させる等のマルウェアが急速に増加しており, 2012 年 12 月には累計 35 万個に達している [2]. この利用者情報には, 端末の電話番号や電話帳のデータ等がある. そのため, マルウェアのインストールを防ぐ仕組みが非常に重要となっている。

Android にはマルウェアのインストールを防止するためのパーミッションシステムが存在する. このシステムはアプリケーションによる端末内の利用者情報へのアクセスやセキュリティに関わる操作を制限する. たとえば, インターネットへのアクセスや, 連絡先データの読み取りのパーミッションがある. この操作を行う場合には, アプリケーション内にパーミッションを記述する必要がある. 記述されたパーミッションはインストール時にユーザへ通知され, ユーザがインストールするかを選択することができる。

しかし, 当該システムはパーミッションが何の目的で利用されているかを通知しないため, ユーザがパーミッションの利用目的を理解しないままマルウェアをインストールして利用者情報を漏えいしてしまうという問題が生じている. マルウェアは利用者情報を取得するパーミッションとインターネットを利用するパーミッションを組み合わせることで利用者情報を漏えいすることから, これらのパーミッションに関係があることを可視化すれば, ユーザはインストール前にマルウェアを判断しやすいと考えられる。

本論文では, アプリケーションが利用しているパーミッションが何の目的で利用されているかをユーザがより判断しやすいように, Android のアプリケーションファイルである apk ファイルからパーミッションの関係を解析し, パーミッションの利用目的を可視化する方法を提案する。



図 1 パーミッションのユーザへの通知

Fig. 1 The notice of permissions to a user.

## 2. Android パーミッション通知の問題と研究目的

### 2.1 利用者情報

総務省から, 利用者情報の取扱いに関する対応がまとめられたスマートフォン プライバシー イニシアティブ [1] が公表されている. 表 1 は, スマートフォン プライバシー イニシアティブであげられている利用者情報から, パーミッションシステムで制限されている情報を抜粋したものである. たとえば, スマートフォンには第三者の情報として, 電話帳で管理されるデータが格納されており, 氏名, 電話番号, メールアドレス等の利用者情報が含まれる。

### 2.2 利用者情報漏えい対策に関する技術や既存研究

Android 端末には, 電子メールや電話帳など, 多くの利用者情報が格納されている. アプリケーションがこの情報にアクセスする場合はパーミッションが必要であり, それは図 1 のようにユーザへ通知される. 図 1 左のリストの中に「完全なインターネットアクセス」の項目があるが, これはインターネットの通信を行うための INTERNET パー

\*1 総務省, スマートフォン プライバシー イニシアティブ, p.44 の図表 4-2 をもとに著者が手を加え作成した。

ミッションが使用されることを表す。図 1 右は、図 1 左の「完全なインターネットアクセス」を選択した際に表示される画面であり、より詳細な情報を提供している。ユーザはこのパーミッション通知を目視し、安全なアプリケーションであるかを判断してインストールするかを決定する。

これまでに、情報漏えいの可能性をユーザに通知する研究や技術開発はさかに行われている [3], [4], [5], [6], [7]。たとえば、宣言されたパーミッションからのリスク判断 [3], [4] や、利用者情報またはパーミッションに係る動作時のユーザへの許可確認 [5], [6], [7] がある。プログラムの実行が必要な方法 [5], [6], [7] では、正確な情報が取得可能な反面、許可を判断するユーザに必ずしも十分な知識があるとは限らないという問題や、急速に増加するアプリケーションを第三者が漏れなく効率的に検査することが困難であるという問題がある。その相補として、本研究ではコードを静的に解析することにより情報漏えいの可能性を可視化し、多くのユーザ間での情報共有に焦点を当てる。

コードを解析する方法 [3], [4] では、プログラムの実行順序までを解析しない。ゆえに、利用者情報取得に必要なパーミッションと、外部へ通信するパーミッションが関係することによって生じる現象は不明である。また、コードを静的に解析する方法では、パーミッションとこれを必要とする API の呼び出し (API call) の対応付けが不可欠である。その対応付けとして既存の Permission Map [8] がある。しかし、1つのパーミッションで取得できる利用者情報は複数存在しているが、当該 Map に利用者情報は明示されておらず、不十分である。たとえば、READ\_PHONE\_STATE パーミッションにより IMEI や端末の電話番号が取得できるが、パーミッションの粒度では個別の利用者情報の特定まではできない。

以上から、本研究では下記の問題の改善を図る。

- パーミッションの関係により生じる現象が不明である。
- パーミッションの粒度では個々の利用者情報の特定には不十分である。

初めに、パーミッションの関係により生じる現象が不明である問題について説明する。たとえば、アプリケーション内で「完全なインターネットアクセス」と「連絡先データの読み取り」のパーミッションが記述されていた場合、このパーミッションの組合せは電話帳のデータを外部に送信することが可能である。しかし、実際に電話帳のデータを外部に送信するかどうかはユーザに通知されない。これにより、ユーザは危険性を過大評価して、安全なアプリケーションのインストールをやめてしまう可能性がある。

次に、パーミッションの粒度では個々の利用者情報の特定には不十分であるという問題について説明する。たとえばアプリケーション内で「電話のステータスと ID の読み取りの許可」のパーミッションが記述されていた場合、端末の固有 ID (IMEI)、電話番号などのデータを取得するこ

とができる。しかし、どのデータが実際に取得されているかをユーザが判断できる通知はない。そのため、端末を識別するために端末の固有 ID を取得するアプリケーションが、本来必要としない電話番号を扱っていてもユーザは知ることができない。これにより、ユーザは危険性の高いアプリケーションを見過ごす可能性がある。

そこで本論文では、アプリケーションが利用しているパーミッションが何の目的で利用されているかをユーザがより判断しやすいように、パーミッションが関係することによって生じる現象や、各パーミッションによって取得される利用者情報をアプリケーションから解析・可視化する方法を提案する。

### 3. 提案手法

本論文では、Java コードで記述されたアプリケーションのコードを解析範囲とした際に、目的を達成できるかどうかを見極めるために Android アプリケーションの静的解析方法を試案する。なお、Android SDK のライブラリのソースコードやネイティブコードは解析範囲としていない。

この Android SDK は、Google が公開している Android アプリケーションの開発環境であり、コンパイラ、ライブラリ、ドキュメント等が含まれる。ライブラリについては、処理の実装がされていないスタブとなっている。本手法では、Android の API を識別するために、このライブラリを用いる。

#### 3.1 アプリケーションの静的解析

初めに提案手法を説明するための事例を説明する。図 2 は端末の電話番号を外部のサーバへ送信するマルウェアのコード片である。このコードでは、getLineNumber() メソッドによって取得された端末の電話番号が変数 tel に格納され、その格納されている電話番号が post\_params, httpPost の順番で渡されて、execute() メソッドで外部へ送信されている。

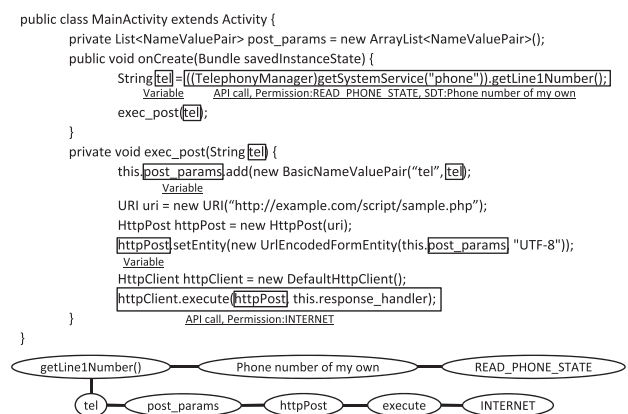


図 2 パーミッション間の関係の解析

Fig. 2 The analysis of the relation of the permissions.

表 2 API call とパーミッションの対応関係の一部

Table 2 A part of mapping between API calls and permissions.

| API call                                            | パーミッション                             | 取得データの種類 |
|-----------------------------------------------------|-------------------------------------|----------|
| java.net.URL.getContent()                           | android.permission.INTERNET         | —        |
| android.telephony.TelephonyManager.getLine1Number() | android.permission.READ_PHONE_STATE | 端末の電話番号  |
| android.bluetooth.BluetoothDevice.getName()         | android.permission.BLUETOOTH        | —        |

次に提案手法の概要を説明する．解決すべき課題は次の3点と考えられる．(1) 利用者情報が取得される箇所を特定できるか，(2) プログラム内の変数等を介した利用者情報の流れが追えるか，(3) 利用者情報が外部へ送信されることを特定できるか．

(1) については，特定の API には呼び出しに必要なパーミッションがあり，利用者情報の取得に必要である．たとえば `getLine1Number()` は `READ_PHONE_STATE` パーミッションを必要とする．同様に特定の API には，外部と通信するためのパーミッションを必要とするものがある．たとえば `execute()` は `INTERNET` パーミッションを必要とする．また，API の呼び出し方により取得できる利用者情報，つまり Sensitive Data Type (以後，単に SDT) が決まる．たとえば `getLine1Number()` では SDT として端末の電話番号が取得される．このようにアプリケーションによって変わらない，API と SDT とパーミッションの静的な関係の紐付けは 3.1.1 項で詳説する．

(2) については，静的解析で端末の電話番号の取得から外部へ送信されるまでの流れにおける変数や API call を紐付ける．図 2 の四角で囲われている要素は解析中に紐付けられる要素であり，たとえば `tel`，`httpPost`，`post_params` は変数を表しており，`getLine1Number()` と `execute()` は API call を表す．このようにアプリケーションによって変わる，静的な関係の紐付けは 3.1.2 項で詳説する．

(3) については，上記の紐付けにより，図 2 下部の関係が抽出でき，`READ_PHONE_STATE` パーミッションによって端末の電話番号が取得され，`INTERNET` パーミッションによって外部へ送信されていることが判断できる．このような紐付けの解釈方法を 3.1.3 項で詳説する．

提案手法では「API・SDT・パーミッションの紐付け」，「変数・定数・API call・引数の紐付け」，「外部へ送信する情報の特定」の手順で解析を行う．次項より各手順の詳細な説明について述べ，続けて Android 特有の課題である「隠蔽されているメソッドへの対処」，「インテントへの対処」についてと対処法について述べる．

### 3.1.1 API・SDT・パーミッションの紐付け

API call とパーミッションを紐付けるためには，パーミッションを必要とする API call を特定する必要があるため，API call とパーミッションの対応関係のデータが必要となる．この対応関係は Android SDK 公式のリファレンスにもデータがあるが，このデータには不足や誤りがあ

```
ContentResolver cr = getContentResolver();
Cursor cursor = cr.query(ContactsContract.CommonDataKinds.Email.CONTENT_URI,
                        null, null, null, null);
int index = cursor.getColumnIndex(ContactsContract.CommonDataKinds.Email.DATA1);
String sdt = cursor.getString(index);
```

図 3 電話帳データの取得

Fig. 3 Collection of contact data.

る [8]．そのため提案手法では，公式リファレンスより正確な対応関係のデータである Permission Map [8] を使用する．なお，Permission Map は Android 2.2 を対象としている．

表 2 は Permission Map の例である．たとえば，`java.net.URL.getContent()` の API を呼び出す場合には `INTERNET` パーミッションを必要とすることを表している．また，提案手法では，最終的にどのような利用者情報が取得されているかを可視化できるように，この対応関係に SDT を新たな要素として追加している．この SDT は，表 2 の取得データの種類の列に記載する．この要素は本研究で独自に調査した内容であり，たとえば `android.telephony.TelephonyManager.getLine1Number()` のような，SDT が一意に定まる API call に対して対応関係を持たせている．この取得データの種類の名称は，Android SDK 公式リファレンスを参考につけたものである．なお，取得データの種類の列は，1 回の API call でデータが得られるもののみを対象としており，複数回の API call でデータが得られるものは対象としていない．たとえば，電話帳では複数回の API call を経て，初めてメールアドレスや電話番号等の詳細なデータを取得できる．これは，電話帳のデータがリレーショナルデータベースのように格納されているため，取得したいデータの URI をもとにデータベースのカーソルを取得し，そのカーソルの行の特定列を取得することで，データを取得できる構造となっているためである．具体例として，図 3 に電話帳からメールアドレスを取得するためのコード片を記載する．変数 `sdt` には，電話帳から取得したメールアドレスが格納されるが，それには `getContentResolver`，`query`，`getColumnIndex`，`getString` と複数回の API call を行う必要がある．しかし，この手続きでは `READ_CONTACTS` パーミッションを必要とするため，電話帳のデータが使用されていることは識別できる．

### 3.1.2 変数・定数・API call・引数の紐付け

利用者情報の流れを追えるように，解析時に行う紐付けについては，「変数と定数」，「変数と変数」，「変数と API

表 3 紐付けの例

Table 3 The mapping of source code elements.

| 紐付ける箇所         | 紐付ける場合の例                                                    |
|----------------|-------------------------------------------------------------|
| 変数と定数          | param = "http://example.com/";                              |
| 変数と変数          | tmp = param;                                                |
| 変数と API call   | param = url.getContent();                                   |
| API call とその引数 | url.getContent(param);                                      |
| 仮引数と実引数        | getContent(param);<br>private void getContent(String arg){} |

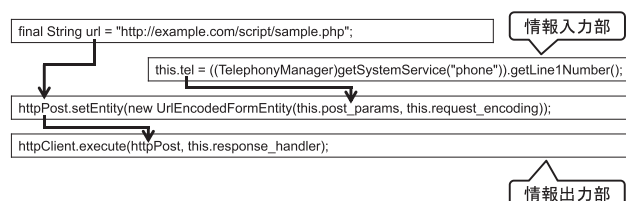


図 4 データの流れ

Fig. 4 The flow of the data.

call], 「API call とその引数」, 「仮引数と実引数」に関して紐付けを行う。「変数と API call」, 「API call とその引数」に関しては, API call が表 2 の対応関係で定義されている場合, パーミッションや SDT の紐付けも行う。

表 3 はコード片による例である。「変数と定数」では変数 param と定数 http://example.com/ を, 「変数と変数」では変数 tmp と変数 param を, 「変数と API call」では変数 param と getContent() メソッドの完全修飾名を, 「API call とその引数」では getContent() メソッドの完全修飾名と引数 param を, 「仮引数と実引数」ではメソッド実行時の実引数 param とメソッドの仮引数 arg を各々紐付ける。

### 3.1.3 外部へ送信する情報の特定

外部へ送信する情報は, 情報入力部によって得られたデータが情報出力部で用いられているかどうかで特定される。情報入力部とは, パーミッションによって端末内の情報を取得できる API call のことであり, 具体的には電話番号の取得や電話帳のデータの取得などがある。また, 情報出力部は, パーミッションによって外部にデータを送信することができる API call のことであり, INTERNET パーミッションまたは SEND\_SMS パーミッションを必要とする API call と 3.1.5 項で詳述するインテントがある。

図 4 は情報入力部と送信先 URL の記述部から情報出力部までのデータの流れである。3.1 節で述べた関係の抽出により, 送信先 URL を含む変数 url が httpPost と紐付き, 情報入力部で取得された端末の電話番号を含む変数 this.tel が this.post\_params と紐づく。さらに this.post\_params と httpPost が紐付き, 最終的に情報出力部である execute() メソッドの実引数として httpPost が用いられているため, 情報入力部で取得した情報が情報出力部によって外部へ送信されていることが特定できる。

### 3.1.4 隠蔽されているメソッドへの対処

Android SDK のクラスを継承したサブクラスの中には, 呼び出し順序が隠蔽されているメソッドがある。図 5 は Android SDK の AsyncTask クラスを継承したサブクラスの実行順序を簡単に表した図である。SampleActivity クラスの onCreate() メソッドにおいて, SampleTask 型オブジェクトである task を生成して, そのオブジェクトの execute() メソッドを実行する。このとき, SampleTask クラスは AsyncTask クラスを継承したサブクラスのため, 実際には AsyncTask クラスの execute() メソッドが呼び出される。この execute() メソッドではサブクラスでオーバーライドされている onPreExecute(), doInBackground(), onPostExecute() メソッドを順番に呼び出している。

onPreExecute(), doInBackground(), onPostExecute() メソッドはアプリケーション開発者の定義したコードが含まれるため, 解析は必須である。しかし, 図 5 の AsyncTask クラスは説明上簡単な構造となっているが, 実際には他のクラスの呼び出しなど複雑な構造となっているため解析が難しい。そこで提案手法では AsyncTask の execute() メソッドである, android.os.AsyncTask.execute() が呼び出される場合, アプリケーション開発者が定義しているサブクラスの onPreExecute(), doInBackground(), onPostExecute() メソッドをこの順番で解析するようにハードコーディングで対処を行っている。

### 3.1.5 インテントへの対処

Android にはインテントと呼ばれる, アプリケーション内の画面遷移や別のアプリケーションの起動を行うことのできる仕組みがある。このインテントはデータの送受信が可能のため, 別アプリケーションとのデータの送受信が可能である。そのため, アプリケーションの組合せによって外部へ利用者情報が送信される可能性がある。たとえば, アプリ A は READ\_PHONE\_STATE パーミッションのみを記述しており, アプリ B は INTERNET パーミッションのみを記述している。それぞれのアプリケーションのインストール時にはパーミッションの組合せがないため, 一見安全なアプリケーションであると考えられる。しかし, アプリ A が端末の電話番号を取得し, インテントでその電話番号をアプリ B が受信して外部に送信することが可能である。また, アプリ B がインテントでアプリ A を起動し, 戻り値として電話番号を受信して外部に送信することが可能である。

インテントへの対処では, アプリケーション内の画面遷移では遷移先のクラスを解析し, 他のアプリケーションの起動では情報の入出力部として API call の解析を行う。

### 3.1.6 動的アプリケーションに対する静的解析の限界

提案手法では静的解析を用いている。静的解析ではアプリケーションのコードをもとに解析するため, コードを網羅することが容易である。しかし, Java のリフレクション

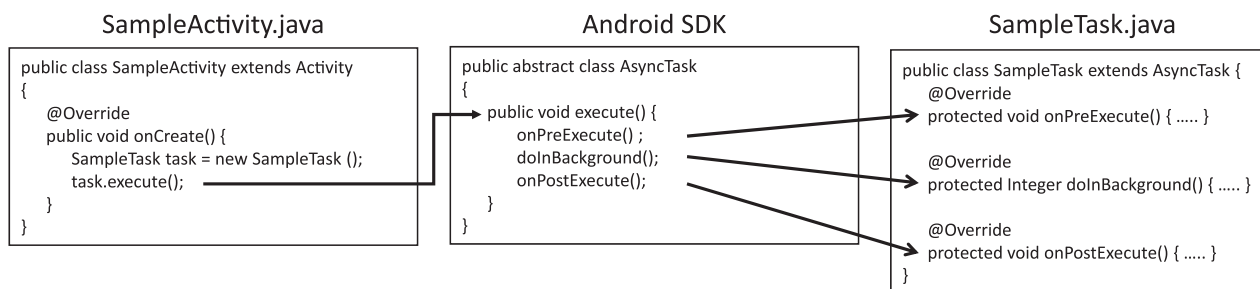


図 5 AsyncTask クラスを継承したサブクラスの実行順序

Fig. 5 The order of analysis for the AsyncTask class.

や、外部のコードを実行する動的クラスローダ等、アプリケーションの実行時に呼び出されるコードが決定される動的アプリケーションの場合、静的解析では当該のコードを正確に解析することができない。また、アプリケーション内にブラウザを埋め込む WebView についても、実行時に外部からコードが取得される。静的解析では WebView への操作を情報出力部として扱うことで識別可能であると考えられるが、WebView に表示されるコンテンツの振舞いは解析できないため、本来漏えいしていない利用者情報を漏えいしたと誤検知する原因となりうる。動的解析ではこれらの解析が可能のため、動的アプリケーションを正確に解析するためには動的解析を用いる必要がある。動的解析の導入については今後の課題とする。

### 3.2 パーミッションの利用目的の可視化

3.1 節の手法で抽出したパーミッション間の関係をもとにパーミッションの利用目的の可視化を行う。図 6 は図 2 のコード片に対する可視化結果である。情報入力部と出力部のパーミッションに関連があるため、「完全なインターネットアクセス」と「電話のステータスと ID の読み取り許可」のパーミッションの組合せによって、端末の電話番号が特定の URL に送信されている可能性があることを表示する。外部送信に関わる文字列に関しては、データを外部送信する場合、ソースコードに文字列として送信するデータの識別子が書かれている場合がある。そのため、ユーザの判断材料の 1 つとして、外部送信を行う動作において関連している文字列を表示する。また、送信元コードの種別に関しては URL が広告に用いるものであった場合、「広告」と表示を行う。これは、外部送信されるデータがどのようなコードによって送信されているかをユーザに伝え、判断材料の 1 つとして扱えるようにするためである。なお、広告の判断には、広告に用いられる URL を記述したリストが別途に必要となり、可視化時にそのリストを入力として与える。現状では広告のみ識別可能であるが、ほかにも利用統計やクラッシュレポート等の情報収集モジュールが考えられる。これらのモジュールへの対応については今後の課題とする。

| 完全なインターネットアクセス(android.permission.INTERNET)<br>電話のステータスとIDの読み取りの許可(android.permission.READ_PHONE_STATE) |                                                               |
|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| 外部送信の可能性                                                                                                | 端末の電話番号                                                       |
| URL                                                                                                     | http://example.com/script/sample.php                          |
| 外部送信に関わる文字列 ①                                                                                           | UTF-8<br>http://example.com/script/sample.php<br>phone<br>tel |
| 送信元コードの種別 ①                                                                                             | 不明                                                            |

図 6 可視化結果

Fig. 6 The result of highlighting.

| ソースコード                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> public void sampleMethod(Bundle bundle) {     exec_post(this.androidid, this.tel, this.data); } </pre>                                                                                                                                                                                                                                                                                                                                                                                |
| soot                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <pre> \$R0 := @this: com.example.sample.SampleClass \$R1 := @parameter0: android.os.Bundle \$R4 = \$R0.&lt;com.example.sample.SampleClass: java.lang.String androidid&gt; \$R2 = \$R0.&lt;com.example.sample.SampleClass: java.lang.String tel&gt; \$R3 = \$R0.&lt;com.example.sample.SampleClass: java.lang.String data&gt; specialinvoke \$R0.&lt;com.example.sample.SampleClass: void exec_post (java.lang.String,java.lang.String,java.lang.String)&gt;(\$R4, \$R2, \$R3) return </pre> |

図 7 ソースコードと soot のコードの比較

Fig. 7 The difference between source code of an application and the code in soot.

## 4. 評価

提案手法によって正しい解析結果を得ることができるかを確認するために、試作したツールを用いて評価を行った。このツールは Java バイトコードの最適化ツールである soot [9] を用いており、アプリケーションの apk ファイルを入力として、図 6 の可視化結果の HTML を出力するものである。そして、その結果に基づいて、ユーザがパーミッションの利用目的から不正アプリケーションをどのように判断できるかを考察した。なお、soot の出力するコードは図 7 のようにソースコードとは異なるが、3.1.2 項で述べた紐付けを行う箇所は soot の出力するコードにも現れるため、3.1 節の手法で解析が可能である。

| 完全なインターネットアクセス(android.permission.INTERNET) |                                   |
|---------------------------------------------|-----------------------------------|
| URL                                         | http://example.com/rss/sample.xml |
| 外部送信に関わる文字列 ①                               | http://example.com/rss/sample.xml |
| 送信元コードの種別 ①                                 | 不明                                |

図 8 RSS 取得アプリケーションの可視化結果

Fig. 8 The result of highlighting of the RSS application.

#### 4.1 適用対象のアプリケーション

試作したツールの適用対象として、24 個の正規アプリケーションと 6 個のマルウェアを利用した。正規アプリケーションはマルウェアでないアプリケーションであり、マルウェアは READ\_CONTACTS パーミッションを要求するアプリケーション群から選択したものである。このマルウェアは、モバイル端末のマルウェアを収集している contagio mobile [10] より収集した。

なお、正規アプリケーションにおいて、利用者情報を外部送信するアプリケーションが 3 件含まれている。外部送信するアプリケーションが少ない理由は、外部に利用者情報を送信するとマルウェアと見なされやすいため、全体的に少ないからである。そのため、その正規アプリケーションとマルウェアをあわせて解析し、外部に利用者情報を送信することを正しく解析できるかを評価する。

また、可視化結果が正しいか判断するために、正規アプリケーションでは secroid [4] との比較を行い、一部アプリケーションはソースコードで目視による情報入力部と出力部の間の関係性を確認した。マルウェアについては、dex2jar [11] と JD [12] を用いて逆コンパイルしたソースコードを用意し、正規アプリケーションのソースコードによる確認と同様に行った。

#### 4.2 可視化結果

外部に利用者情報を送信していない 21 個のアプリケーションは、すべて正しく解析できた。たとえば、図 8 は RSS 取得アプリケーションを対象とした可視化結果である。情報出力部として INTERNET パーミッションを利用しているが、情報入力部のパーミッションを利用していないため、外部送信の可能性は表示されていない。また、接続先の URL としてソースコード中に埋め込まれている RSS の URL が取得できていることが分かる。この結果より、対象としたアプリケーションが RSS の取得のみに INTERNET パーミッションを利用していて、端末のデータを外部に送信していないため安全なアプリケーションであると判断できる。

外部に利用者情報を送信している 9 個のアプリケーションのうち、7 個のアプリケーションを正しく解析できた。たとえば、図 9 は the movie 系のアプリケーションを対象とし

| 完全なインターネットアクセス(android.permission.INTERNET)<br>電話のステータスとIDの読み取りの許可(android.permission.READ_PHONE_STATE)<br>連絡先データの読み取り(android.permission.READ_CONTACTS) |                                                                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| 外部送信の可能性                                                                                                                                                 | 端末の電話番号<br>電話帳のデータ                                                                       |
| URL                                                                                                                                                      | http://[redacted].jp/get51.php                                                           |
| 外部送信に関わる文字列 ①                                                                                                                                            | id<br>UTF-8<br>android_id<br>http://[redacted].jp/get51.php<br>1<br>phone<br>tel<br>data |
| 送信元コードの種別 ①                                                                                                                                              | 不明                                                                                       |

図 9 the movie 系のアプリケーションの可視化結果\*2

Fig. 9 The result of highlighting of the malware.

た解析結果である。the movie 系のアプリケーションはマルウェアであり、動画再生アプリと紹介されている。このアプリケーションは、INTERNET・READ\_PHONE\_STATE・READ\_CONTACTS パーミッションを要求し、インターネット上の動画を再生するとともに端末の電話番号や電話帳のデータを外部のサーバへ送信する。図 9 では、情報入力部のパーミッションとして READ\_PHONE\_STATE パーミッションと READ\_CONTACTS パーミッション、情報出力部のパーミッションとして INTERNET パーミッションが利用されており、情報入力部と情報出力部に関連があることが表示されている。さらに READ\_PHONE\_STATE パーミッションを利用する getLineNumber() メソッドは端末の電話番号を取得するメソッドのため、外部送信の可能性として「端末の電話番号」が出力されている。READ\_CONTACTS パーミッションについては、API call からでは電話番号やメールアドレスといった詳細な利用者情報が識別できないため、外部送信の可能性としては電話帳全体である「電話帳のデータ」が出力されている。また、接続先 URL についてもソースコード中に埋め込まれている URL が取得されていることが分かる。この結果より、対象としたアプリケーションは動画再生アプリと紹介されているにもかかわらず、READ\_PHONE\_STATE パーミッションを用いて取得された端末の電話番号や、READ\_CONTACTS パーミッションを用いて取得された電話帳のデータが INTERNET パーミッションによって外部に送信されていることが分かるため、アプリケーションの紹介内容と比較することで、マルウェアの可能性が高いことが判断できる。

なお、2 個のアプリケーションで外部に利用者情報を送信していることを正しく解析できなかった。これはツールに不具合が散在したことで、3.1.1 項で述べた複数回の API call を経て初めて特定できるパーミッションが含まれており、それが正しく解析できなかったためである。

\*2 マルウェアに関わる URL のため一部加工した。

### 4.3 考察

RSS 取得アプリケーションや the movie 系のアプリケーションでは、具体的に送信されているデータを表示することによって、図 1 のような Android 標準のパーミッションシステムと比較してパーミッション利用目的の理解の容易さが改善されていることが期待される。

the movie 系のアプリケーションには 3.1.4 項で述べた AsyncTask が利用されており、AsyncTask を継承したサブクラスの解析順序はハードコーディングによって対処を行っているが、一般性が低いため、Android SDK のライブラリのソースコードを解析する等の対処を行う必要がある。

また、適用対象のアプリケーション数が限定的であるため、今回の結果が十分に一般的たりえない可能性がある。そのため、引き続き多くのアプリケーションを解析する必要がある。

## 5. 関連研究

Android 端末にインストールされているアプリケーションが危険なパーミッションの組合せを持っていないか確認することができる Android アプリケーションとして tSpy-Checker [3] がある。このアプリケーションは、端末内の各アプリケーションに記述されているパーミッションを抽出し、パーミッションの組合せから情報漏えいの可能性を提示するものである。しかし、提示する基準は記述されているパーミッションの組合せのみとなっており、本当に情報漏えいが行われているか判断できない。また、インストール済みのアプリケーションのみが対象のため、インストール前に確認を行うことができない。本論文の手法では、インストール前の確認が可能であり、実際にパーミッションの組合せに関係があるかを解析するため、より正確な判断材料をユーザに提供できると期待される。

既存研究には、アプリケーションの実行中にアプリケーションの動作をユーザが制御する手法 [5], [6], [7] がある。

矢儀ら [5] は、SELinux をベースにして開発されている SEAndroid におけるユーザの判断支援や負荷軽減の手法を提案しており、アプリケーションがパーミッションを必要とする API やインテントを利用する場合、ユーザがその動作の許可か拒否を決定することができる。

林ら [6] は、アプリケーションが利用者情報や端末の動作に関連する動作を実行する場合、ユーザがその動作の許可か拒否を決定できる手法を提案している。

加藤ら [7] は、パーミッションが必要な動作が生じた際に、その内容や使用される値をユーザに示し、動作の実行の承認を求めるシステムを提案している。

しかし、これらの研究 [5], [6], [7] はいずれもアプリケーションの実行中にユーザへ特定の動作の許可を求めるものであり、これらの手法を効果的に利用するためにはユーザがパーミッションに関する詳しい知識を持っている必要

があると考えられる。さらにこれらはカスタマイズされた OS のため既存端末への普及は困難である。本論文の手法では、Android 上で動作するシステムではないため OS のカスタマイズも不要である。また、パーミッションに関する知識を持っていないユーザに対しても、外部に送信されているデータの具体的な種類を提示することで従来より判断しやすいと考えられる。しかし、Android ユーザの振舞いの研究 [13] において、ユーザはセキュリティダイアログに対して OK を選択してしまうことが述べられている。このことから、本論文の手法で可視化した結果をユーザが閲覧しないままインストールを続行してしまうという問題が考えられる。この問題への対処については今後の課題とする。

tSpyChecker [3] や研究 [5], [6], [7] はアプリケーションのインストール後に適用可能な手法であるが、本論文の手法と同様にインストール前にアプリケーションの危険性を確認できるサービスに secroid [4] がある。secroid は Google Play に公開されているアプリケーションを解析し、DANGER・HIGH・MID・LOW・SAFE の 5 段階のリスクレベルに分類する。リスクレベルでの表示はユーザにとって直感的に理解しやすいが、このレベル判定は電話帳のデータの利用やサーバソケットの利用といった基準で行われているため、正当な目的でパーミッションを利用しているアプリケーションでもリスクレベルが高く判定されてしまう可能性がある。本論文の手法では、アプリケーションの挙動に関して情報漏えいの観点から、情報入力部で得られるデータの流れを解析し、パーミッションどうしに関連があるかなど、よりきめ細かい判断材料をユーザに提供することができる。

また、磯原ら [14] は、スマートフォン プライバシー イニシアティブで策定されているプライバシーポリシーの作成支援機構を提案している。本論文の手法では、アプリケーションが利用者情報を外部に送信しているかを判定できるため、このプライバシーポリシーの作成支援に活用できるのではないかと考えられる。

IPA では、Android アプリの脆弱性に関するレポート [15] を公開しているが、その中ではパーミッションのみでは判断し難いマルウェアの事例があげられている。たとえば、SD カードからのデータ読み取りにはパーミッションが不要であり、INTERNET パーミッションのみで、SD カード内のデータを外部に送信できる状況があげられている。本研究では、Permission Map を活用した静的な解析法を提案しているため、パーミッションに係る API call を情報入出力部として判定している。しかし、本論文の手法では、情報入出力部と見なす API call を決定すれば利用可能であり、パーミッションによらない解析も可能である。そこで、File の生成や読み書きを行う API を情報入力部として扱うことで、ファイルやその内容を外部に送信する可能性

を指摘できることが期待できる。

## 6. おわりに

### 6.1 まとめ

本論文では、静的解析によりアプリケーションの利用目的を可視化する手法を提案した。そして、Android アプリケーションを静的解析した場合の課題と対処方法について述べた。また、提案手法に基づくツールの試作を行い、実際にアプリケーションに適用を行い、マルウェアの判断に有用であることが期待されるという結果を得ることができた。

### 6.2 今後の課題

今後の課題としては、Permission Map の追加調査、取得データの種類の調査、送信元コードの種別の調査、Android SDK の解析、評価、動的解析の導入、未利用パーミッションの検出、レビューサイトの構築があげられる。

Permission Map の追加調査については、Permission Map が Android 2.2 を対象としているため、2.3 以降で追加または変更された API とその利用に必要なパーミッションの関係を調査する。

取得データの種類の調査については、3.1.1 項で述べた API call と取得データの種類の関係を調査し、対応関係を充足させる。また、電話帳の名前や電話番号といった、詳細な利用者情報を識別するための手法を検討する。

送信元コードの種別の調査については、3.2 節で述べたように送信元コードの種別が現状では広告のみの識別のため、利用統計やクラッシュレポートなどのモジュールを識別できるようにする。この識別は、モジュールのパッケージ名と種別の組を記述したリストを用意し、解析時に情報出力部のクラスのパッケージ名と照合することで実現を図る。

Android SDK の解析については、4.3 節で述べたようにハードコーディングでは一般性が低いため、Android SDK のライブラリのソースコードを解析する方法を検討し、Android のバージョンアップが行われても汎用的に提案手法を活用できるようにする。

評価については、さらに多くのアプリケーションに対して提案手法を適用し、解析と可視化の正確性を評価する。また、提案手法によって Android 標準のパーミッションシステムより、ユーザにとってマルウェアであるかの判断が容易となるかどうかを、従来と提案手法それぞれの情報表示でユーザの判断が変わるかどうかの実験を行う等して評価する。

動的解析は、3.1.6 項で述べたとおり、アプリケーション実行時に行われるリフレクションや動的クラスローダ、WebView といった静的解析では得ることができない情報を得るために必要である。しかし、Android を対象とした

未利用のパーミッションが検出されました

ユーザの承認無しで以下のパーミッションに関わる機能が追加できます  
連絡先データの読み取り(android.permission.READ\_CONTACTS)

図 10 未利用パーミッション検出のイメージ図

Fig. 10 An alert for notifying unused permissions.

自動的かつ網羅的な動的解析は難しく、十分な手法がない [16]。そのため、アプリケーションのコードを網羅的に解析する静的解析により、情報漏えいの疑わしい箇所を特定し、その箇所を中心に動的解析を行い、正確な解析を可能にするなど、動的解析手法の検討やその提案手法との関係方法を模索する。

Android アプリケーションでは、未使用のパーミッションでも宣言できる。これにユーザが 1 度でも許可を与えると、アプリケーションのアップデート時に当該パーミッションが使用されるようになった場合でも、承認画面は表示されない。その結果、公開当初は無害を装い、アップデート後にマルウェア化するアプリケーションがあるため、未利用のパーミッション検出は重要である。提案手法では、パーミッションを必要とする API call を特定しているため、Permission Map に定義されている範囲に限られるが、実際に使用されているパーミッションとアプリケーションで宣言されているパーミッションを比較することで、未利用のパーミッションを検出できる。図 10 は実現した際のイメージ図である。このように、未利用のパーミッションの危険性をユーザに提示することで、バージョンアップによる意図しない利用者情報漏えいを避けられる可能性があるため、未利用パーミッションの検出について検討する。

レビューサイトの構築については、Android ユーザの振舞いの研究 [13] において、多くのユーザはパーミッションを注視しないが、レビューは注視するという傾向が見られる。また、パーミッションについて熟知している一部のユーザが不審なパーミッションについてレビューを書くということが述べられている。このことから、提案手法による可視化結果をレビューサイトに組み込むことによって、パーミッションについて熟知している一部のユーザがパーミッションに関してレビューを書くことを助ける。また、そのレビューを見た多くのユーザがパーミッションの利用目的を理解できる可能性があるため、レビューサイトの構築を検討する。

## 参考文献

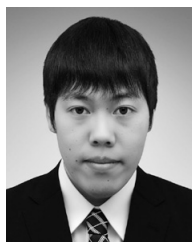
- [1] 総務省：スマートフォン プライバシー イニシアティブ (オンライン), 入手先 <[http://www.soumu.go.jp/main\\_content/000171225.pdf](http://www.soumu.go.jp/main_content/000171225.pdf)> (参照 2014-03-11).
- [2] トレンドマイクロ株式会社：2012 年度インターネット脅威年間レポート (オンライン), 入手先 <<http://www.trendmicro.co.jp/jp/security-intelligence/threat-report/>>

- articles/20130906123029.html) (参照 2014-03-11).
- [3] タオソフトウェア株式会社: Android ソフトウェア - tSpyChecker (オンライン), 入手先 <http://www.taosoftware.co.jp/android/spychecker/> (参照 2014-03-11).
  - [4] ネットエージェント株式会社: Android アプリの潜在リスクチェックは secroid (セキュロイド) (オンライン), 入手先 <http://secroid.jp/> (参照 2014-03-11).
  - [5] 矢儀真也, 山内利宏: SEAndroid の拡張による AP の動的制御手法の提案, *Computer Security Symposium 2012*, pp.130–137 (2012).
  - [6] 林 里香, 後藤厚宏: Android アプリケーション利用の安全性を高めるアプリケーション動作の「見える化」, *Computer Security Symposium 2012*, pp.138–145 (2012).
  - [7] 加藤 真, 松浦佐江子: ユーザの承認によるアプリケーション実行時の Android マルウェア対策, 情報処理学会研究報告 CSEC, Vol.2013-CSEC-61, No.6, pp.1–6 (2013).
  - [8] Felt, P.A., Chin, E., Hanna, S., Song, D. and Wagner, D.: Android Permissions Demystified, *Proc. 18th ACM Conference on Computer and Communications Security (CCS '11)*, pp.627–637 (2011).
  - [9] Lam, P., Qian, F., Lhotak, O. and Bodden, E.: Soot: a Java Optimization Framework (オンライン), 入手先 <http://www.sable.mcgill.ca/soot/> (参照 2014-03-11).
  - [10] Parkour, M.: contagio mobile (オンライン), 入手先 <http://contagiomindump.blogspot.jp/> (参照 2014-03-11).
  - [11] Panxiaobo and yyjdelete: dex2jar (オンライン), 入手先 <https://code.google.com/p/dex2jar/> (参照 2014-03-11).
  - [12] Dupuy, E.: Java Decompiler (オンライン), 入手先 <http://jd.benow.ca/> (参照 2014-03-11).
  - [13] Felt, P.A., Ha, E., Egelman, S., Haney, A., Chin, E. and Wagner, D.: Android Permissions: User Attention, Comprehension, and Behavior, *Proc. 8th Symposium on Usable Privacy and Security (SOUPS 2012)*, No.3, pp.1–14 (2012).
  - [14] 磯原隆将, 川端秀明, 竹森敬祐, 窪田 歩: Android アプリの静的解析を用いた利用 API と外部モジュール検知によるプライバシーポリシー作成支援機構, 情報処理学会研究報告 CSEC, Vol.2013-CSEC-62, No.63, pp.1–8 (2013).
  - [15] IPA: 『Android アプリの脆弱性』に関するレポート (オンライン), 入手先 <http://www.ipa.go.jp/files/000009386.pdf> (参照 2014-03-12).
  - [16] 塩治榮太郎, 秋山満昭, 岩村 誠, 針生剛男: GUI 構造に基づいた Android アプリケーション動的解析支援の検討, *Computer Security Symposium 2012*, pp.36–43 (2012).

## 推薦文

Android アプリのパーミッションの可視化は, アプリをインストールしようとするユーザに有益な情報をもたらす. 本論文はアプリの静的解析による合理的なパーミッションチェック手法を提案しており, 簡便さと検査精度の観点でコストパフォーマンスの良いパーミッションチェックを実現している. 今後の当研究分野の発展に有用であるため, 推薦論文として推薦したい.

(コンピュータセキュリティ研究会主査 西垣正勝)



坂下 卓弥 (学生会員)

2013 年信州大学工学部情報工学科卒業. 現在, 信州大学大学院理工学系研究科情報工学専攻在学.



小形 真平 (正会員)

2012 年芝浦工業大学大学院博士課程修了. 博士 (工学) 取得. 現在, 信州大学大学院理工学系研究科助教. ソフトウェア工学におけるモデル駆動工学やオブジェクト指向開発技法の研究に従事.



海谷 治彦 (正会員)

1994 年東京工業大学博士 (工学) 取得. 現在, 神奈川大学理学部教授.



海尻 賢二 (正会員)

1977 年 3 月大阪大学大学院工学研究科博士課程修了. 工学博士. 1977 年 4 月より信州大学工学部情報工学科に奉職し, 現在, 教授. ソフトウェア工学の研究に従事.