Vol.2013-CSEC-62 No.21
Vol.2013-SPT-6 No.21
2013/7/18

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[†1]　小形真平[†1]　海谷治彦[†1]　海尻賢二[†1]

Android には個人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，マルウェアによる個人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 等に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，アプリケーションのソースコードを静的解析し，利用目的を可視化する方法を提案する．本論文では，提案手法が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．

Static Analysis for Highlighting what Android Application does with Permissions

TAKUYA SAKASHITA[†1]　SHINPEI OGATA[†1]
HARUHIKO KAIYA[†1]　KENJI KAIJIRI[†1]

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.

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，個人や企業が Android アプリ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 の普及に伴い，連絡先データの個人情報を外部に流出させる等のマルウェアが急速に増加している. 図 1 はマルウェアが 1 年間で累計 35 万個まで急増したことを表すグラフである[1]．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 Android 端末内の個人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．例えば，インターネットへのアクセスや，連絡先データの読み込み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を続行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個人情報を漏えいさせてしまう問題が生じている．マルウェアは個人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て個人情報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．

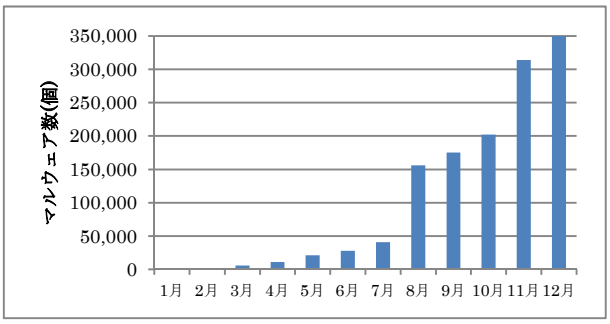

図 1　2012 年の Android のマルウェア数

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

†1 信州大学大学院理工学系研究科
Graduate School of Science and Technology, Shinshu University

Vol.2013-CSEC-62 No.21
Vol.2013-SPT-6 No.21
2013/7/18

しやすいように，アプリケーションのソースコード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．

2. Android パーミッション通知の問題と研究目的

Android 端末には，電子メールや連絡先など，多くの個人情報が格納されている．アプリケーションがこのような情報にアクセスする場合，開発者はアプリケーション内の設定ファイルにパーミッションを記述する必要があり，この記述されたパーミッションはアプリケーションのインストール時にユーザへ通知される．図 2 はアプリケーションインストール時のパーミッション通知の例である．図 2 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，これはインストールするアプリケーション内で INTERNET パーミッションを記述した場合に表示されるものである．このパーミッションを記述することにより，そのアプリケーションはインターネットの通信を行うことが可能となる．図 2 右は，図 2 左において「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元にして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を続行するかを決定する．

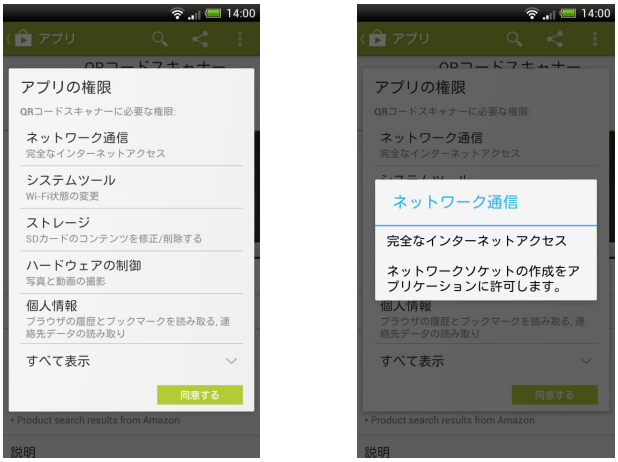

図 2　パーミッションのユーザへの通知

しかし，このパーミッション通知には下記の問題がある．

- パーミッション間の関係の有無がわからない
- パーミッションの粒度が粗い

始めに，パーミッション間の関係の有無がわからない問題について説明する．例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み合わせによって連絡先データを外部に送信することが可能である．しかし，実際に連絡先データが外部に送信されるかどうかまではユーザに通知されない．つまり，これらのパーミッション間の関係の有無はわからない．

次に，パーミッションの粒度が粗い問題について説明する．例えばアプリケーション内で「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，連絡先データに記録されている名前・電話番号・E メールアドレス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，電話をかけるアプリケーションが本来必要としない E メールアドレスを扱っていてもユーザが知ることができない．

どちらの問題も，ユーザがアプリケーションの危険性を過小評価してマルウェアのインストールを続行することや，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中止することにつながると考えられる．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，アプリケーションのソースコードからパーミッション間の関係の有無や各パーミッションによって取得されているデータを解析・可視化する方法を提案する．

3. 提案手法

3.1 アプリケーションのソースコードの静的解析

本節では，パーミッション間の関係や各パーミッションによって取得されるデータをアプリケーションのソースコードから静的解析する手法を提案する．

始めに提案手法の全体像を説明する．図 3 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，getLine1Number()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()メソッドにて外部へ送信されている．

特定の API には，呼び出しに必要なパーミッションがあり，個人情報の取得に必要となる．例えば getLine1Number()では READ_PHONE_STATE パーミッションが必要である．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．例えば execute()では INTERNET パーミッションが必要である．また，API の呼び出し方により取得できる個人情報，つまり Sensitive Data Type（以後、単に SDT）が決まる．例えば getLine1Number()では SDT として端末の電話番号が取得される．このようなアプリケーションに依らず変わらない 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 項にて詳説する．

静的解析においては，端末の電話番号の取得から外部のサーバへ送信されるまでの流れにおける変数や API の呼び出し（API call）を紐付ける．図 3 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，例えば tel, httpPost, post_params は変数を表しており，getLine1Number()と execute()は API call を表す．このようなアプリケーション

に依り変わる静的な関係の紐付けは 3.1.2 項にて詳説する．

上記の紐付けにより，図 3 下部の関係を抽出することができ，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のサーバ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 項にて詳説する．

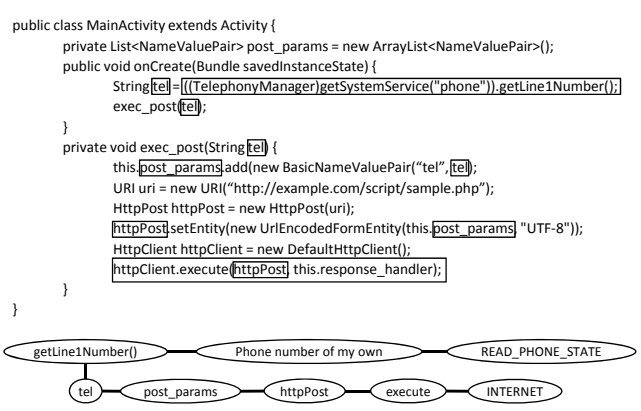

図 3　マルウェアのコード片と静的解析イメージ

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」の手順にて解析を行う．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」についての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 とパーミッションと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る[3]．そのため本論文の手法では，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[3]を使用する．

表 1. API call とパーミッションと SDT の対応関係

API call	パーミッション	SDT
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	
android.accessibilityservice.AccessibilityService.clearWallpaper()	android.permission.SET_WALLPAPER	

表 1 の API call とパーミッションは Permission Map によって規定されている．例えば，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，本論文の手法ではこの対応関係に SDT を新たな要素として追加している．この要素は本研究にて独自に調査した内容であり，例えば android.telephony.TelephonyManager.getLine1Number()のような，得られるデータの種類が一意に定まる API call に対して対応関係を持たせている．なお，SDT は Android SDK 公式リファレンスを参考に名付けたものである．

3.1.2 変数・定数・API call**・引数の紐付け**

静的解析時に，「変数と定数」,「変数と変数」,「変数と API 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 がパーミッションを必要とする場合，3.1.1 項の紐付けを利用して，パーミッションや SDT の紐付けも行う．表 2 は紐付けの例である．「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 をそれぞれ紐付ける．

表 2. 紐付けの例

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); private void getContent(String arg){}

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にて用いられているかどうかで特定を行う．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連絡先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，具体的には 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 項にて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 節で述べたような関係の抽出により，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 と紐付く．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，情報入力部にて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．

図 4　データの流れ

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．この時, 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とになる．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．

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．そこで本論文での手法では execute()メソッドの呼び出し時に，onPreExecute()，doInBackground()，onPostExecute()メソッドの順で解析を行うようにハードコーディングにて対処を行っている．

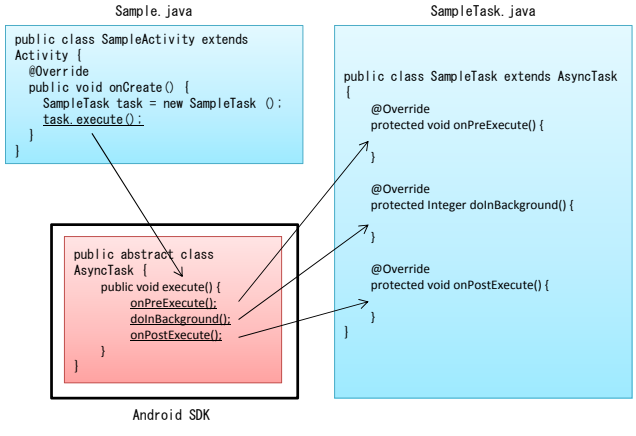

図 5　AsyncTask クラスを継承したサブクラスの実行順序

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なため，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，図 6 のようなアプリケーションの組み合わせによって外部へ個人情報が送信される可能性がある．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み合わせが無いため，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，図 6 上の例ではアプリ A が端末の電話番号を取得し，インテントによって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，図 6 下の例ではアプリ B がインテントによってアプリ 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としては，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．

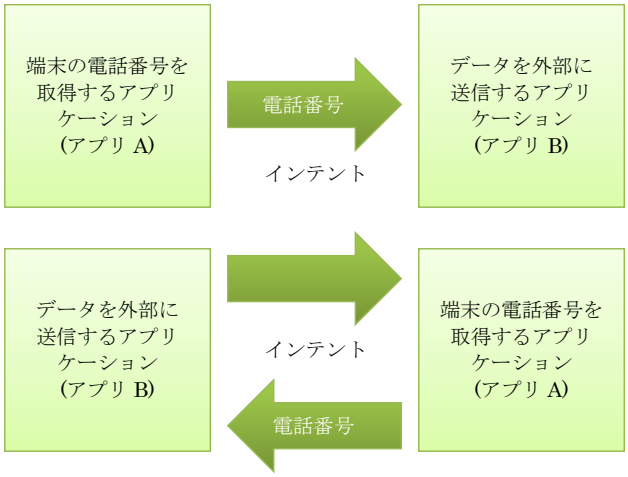

図 6　アプリケーションの組み合わせによる情報流出の例

3.2 パーミッションの利用目的の可視化

3.1 節の手法にて抽出したパーミッション間の関係を元にパーミッションの利用目的の可視化を行う．図 7 は図 3 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み合わせによって，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一つとして，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，目的に関しては URL が広告に用いるものであった場合，「広告」と表示を行う．このために，広告目的で利用される既知の URL をリスト化した広告 URL リストを用意し，広告に用いる URL であるかの判断を行う．これは，外部送信されるデータがどのような目的で送信されているかをユーザに伝え，判断材料の一つとして扱えるようにするためである．

図 7　可視化結果

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 Eclipse の構文解析器である ASTParser を用いており，ZIP 圧縮されているソースコードを入力として，図 7 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．

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，RSS 取得アプリケーションと the movie 系のアプリケーションを利用した．

RSS 取得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，インターネット上の RSS を取得してリスト表示するものである．このアプリケーションは INTERNET パーミッションのみを要求する．

the movie 系のアプリケーションはマルウェアであり，インターネット上の動画を再生すると共に端末の電話番号や連絡先情報を外部のサーバへ送信するものである．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する．今回対象としたアプリケーションはアプリケーション名が「暇つぶし動画」となっているものであり，モバイル端末のマルウェアを収集している contagio mobile[4]より apk ファイルを取得し，dex2jar[5]と JD[6]を用いて逆コンパイルを行い，到達不能コードの除去を行ったソースコードを用意した．到達不能コードについては，dex2jar による逆コンパイルが完全でないために発生しているものである[7].

4.2 可視化結果

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．

図 8　RSS 取得アプリケーションの可視化結果

図 7 は the movie 系のアプリケーションを対象とした解析結果である．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，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 getLine1Number()メソッドは端末の電話番号を取得するためのメソッドのため，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．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，マルウェアの可能性が高いことが判断できる．

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 2 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，必ずハードコーディングされた順序で実行さ

れることの保証を行うことが出来ないため，正しい解析を行うための精度向上に動的解析を導入する必要がある．

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み合わ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 tSpyChecker[8]がある．このアプリケーションは，端末にインストールされているアプリケーションで記述されているパーミッションを抽出し，情報漏えいを行うことが可能なパーミッションの組み合わせを持つアプリケーションを判別するものである．しかし，判別の基準は記述されているパーミッションの組み合わ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することはできない．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間の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．

既存研究としては，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[9], [10]がある．矢儀ら[9]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．また，林ら[10]はアプリケーションが個人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．しかし，これらの研究[9], [10]はどちら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．また，Android ユーザの振る舞いの研究[2]において，ユーザはセキュリティダイアログに対して注視せず OK を選択してしまうことが述べられている．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無いため 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判断が可能であると考えられる．ユーザがセキュリティダイアログを注視しない問題については，6.2 節にて詳説する口コミサイトの導入による解決が考えられる．

tSpyChecker[8]や研究[9], [10]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[11]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，このレベル判定は連絡先情報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，情報入力部にて得られるデータの流れを解析し，パーミッション同士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．

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 アプリケーションのソースコード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のソースコードに適用を行い，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, SDT の調査, ツールの改良, 評価, 動的解析の導入，レビューサイトの構築があげられる．

SDT の調査については, 3.1.1 項で述べた API call と SDT の関係を調査し，対応関係のデータを充足させる．

ツールの改良については，様々なアプリケーションのソースコードに適用させて, 解析可能なコードを拡充させる．

評価については，可視化した情報の正確性の評価と，Android 標準のパーミッションシステムによる情報，従来手法による情報，本手法による情報の 3 点をユーザが閲覧し，実際のユーザが安全なアプリケーションであるかの判断を行うことで本手法の有用性の評価を行う．

動的解析については，アプリケーション実行時行われる動的データバインディングといった静的解析では得ることが出来ない情報を得るために有用である．しかし，Android を対象とした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[12]．そのため，静的解析により可能な限りの解析を行い, 動的解析すべき箇所を限定させる. 例えば, 3.1.4 項で述べた隠蔽されているメソッドへの対処において解析の正確性を向上させるために導入を検討する．

レビューサイトの構築については，Android ユーザの振る舞いの研究[2]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, 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，本手法による可視化結果を口コミ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，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することが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) マイクロトレンド株式会社, 2012 年度インターネット脅威年間レポート, http://jp.trendmicro.com/jp/threat/security_news/monthlyreport/article/20130107041500.html, 2013 年 6 月 13 日
2) Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin and David Wagner, Android Permissions: User Attention, Comprehension, and Behavior, Electrical Engineering and Computer Sciences University of California at Berkeley Technical Report No. UCB/EECS-2012-26, pp.1-14 (2012)
3) Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, David Wagner, Android Permissions Demystified, Proc. of the 18th ACM Conference on Computer and Communications Security, CCS '11, pp.627-637 (2011).
4) Mila Parkour, contagio mobile, http://contagiominidump.blogspot.jp/, 2013 年 6 月 13 日
5) Panxiaobo, yyjdelete, dex2jar, https://code.google.com/p/dex2jar/, 2013 年 6 月 13 日
6) Emmanuel Dupuy, JD | Java Decompiler, http://java.decompiler.free.fr/, 2013 年 6 月 13 日
7) Carlos A. Castillo et al., Android Malware−Past, Present, and Future, http://www.mcafee.com/us/resources/white-papers/wp-android-malware-past-present-future.pdf, 2013 年 6 月 18 日
8) タオソフトウェア株式会社, Android ソフトウェア - tSpyChecker, http://www.taosoftware.co.jp/android/spychecker/, 2013 年 6 月 13 日
9) 矢儀 真也, 山内 利, 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)
10) 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)
11) 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), http://secroid.jp/, 2013 年 6 月 13 日.
12) 塩治 榮太朗, 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)