Vol.2013-DPS-154 No.52
Vol.2013-CSEC-60 No.52
2013/3/15

ContentProvider を用いた利用者情報送信の動的解析手法に関する検討

名雲孝昭† 秋山満昭† 針生剛男†

近年，Android 端末をターゲットとするマルウェアが増加しており，Android マルウェアの対策は重要な課題となっている．Android マルウェアの感染を防止する手法として，アプリケーションマーケットにおける対策は，ユーザへの感染を未然に防ぐだけでなく，マーケット自体の信頼性の担保という観点から重要である．Android マルウェアの中でも，特に利用者情報送信機能を有する Android マルウェアは，昨今社会問題にもなっており，その対策は急務である．そこで，本テーマではマーケット管理者がアプリケーション検査を実施する際に，利用者情報送信を引き起こすアプリケーションを特定する技術の確立を目標とする．これまで，アプリケーションを検査する手法として，アプリケーションの動的解析の手法の 1 つであるテイント解析手法が提案されてきた．既存手法では，対象とする利用者情報が限られていたため，利用者情報送信アプリケーションの特定が困難であった．そこで本テーマでは，利用者情報対象範囲を拡張したテイント解析手法を実装した．また，マーケットのアプリケーションを用いて実験を行い，本手法により利用者情報を外部に送信するアプリケーションを新しく検知した結果を報告する．

Feasibility study on method of dynamic analysis for sending user information using ContentProvider in Android applications

TAKAAKI NAGUMO† MITSUAKI AKIYAMA†
TAKEO HARIU†

Malware that target Android devices is increasing, and countermeasure against them has become an important issue. As one of the approaches for preventing infection of Android malware, taking countermeasure on the application market is important not only for preventing infection to users but also for ensuring reliability of the market itself. Particularly, it is urgent to devise a countermeasure against Android malware causing sending user information. In this research, we aim to establish a technology to extract information to identify applications causing sending user information, intended for use when applications are inspected by a market administrator. To date, as a method for checking applications, taint analysis, which is one of the dynamic analysis methods, has been proposed. Current methods limit the range of user information to be targeted. However, it is difficult to identify applications causing sending user information. In this research, we implemented an extension for the current taint analysis method. In addition, we conducted an experiment with market applications, and report the results showing some newly detected applications that send user information to the outside.

1. はじめに

1.1 背景

近年，Android 端末の普及に伴い，Android を標的とするマルウェア（Android マルウェア）が急増している[1]．一般的に Android マルウェアは，アプリケーションとして，Google が提供する公式マーケットや第三者が提供する非公式マーケットおよび個人 Web サイトを経由して配布されている．端末利用者（ユーザ）は，自身が選択したマーケットから任意のアプリケーションをダウンロードしてインストールすることができるが，マーケット上のマルウェアをインストールしてしまう危険性がある．

Android マルウェアは主に下記の機能を具備する．

- 有料サービスの無断利用
- 端末の管理者権限（root 権限）の奪取
- 利用者情報の送信

有料サービスの無断利用は，プレミアム SMS と呼ばれる課金制の SMS をユーザの承諾なしに送信することなど，金銭的被害を発生させる機能である．例として，2010 年に登場した FakePlayer が挙げられる．端末の root 権限の奪取は，端末の脆弱性を悪用して root 権限を奪取することで，本来保護されているシステム領域を操作する機能である．例として，2011 年に公式マーケットに登場した DroidDream が挙げられる．利用者情報送信機能は，端末の利用者情報をユーザの許諾なく外部サーバへ送信する機能で，近年最も大規模な被害発生の原因となっている．例えば，2012 年 4 月に，「The Movie」と呼ばれるアプリケーション群により数百万件の利用者情報が送信する問題[3]が発生した．また，2012 年 10 月には，「全国電話帳」アプリケーションにより 76 万件の利用者情報が送信する問題[4]が発生した．さらに，アプリケーションに組み込まれた広告モジュールが過剰に利用者情報を収集・送信する問題や，端末識別情報などを

* † NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories.

ユーザの許諾を受けずに第三者に提供する方式（オプトアウト方式）に起因する情報送信も，多くの被害を発生させている[5]．また，ユーザに利用者情報収集アプリケーションのインストールを誘導する，公式マーケットを偽装したマーケットが発生している[6]．

以上から，本研究では，最近最も被害が発生している，利用者情報の送信を引き起こすアプリケーションを特定し，情報送信を防止するための情報を取得する技術の確立を目標とする．

1.2 対策手法

利用者情報の送信を引き起こすアプリケーションを特定する手法は，以下の 2 つに分けられる．

- ユーザ端末におけるアンチウイルスソフトウェアの使用
- マーケット側におけるアプリケーション検査

前者は，ユーザが利用する端末においてアプリケーションを検査し，ユーザ端末上から利用者情報の送信を引き起こすアプリケーションを排除する手法である．しかし，ユーザ端末上におけるアンチウイルスソフトウェアは，他のアプリケーションと同等の限られた権限で動作するため，検査できる範囲に限界がある．後者は，マーケット管理者がアプリケーションを検査し，利用者情報の送信を引き起こすアプリケーションをマーケットから排除する手法である．この手法を適用することで，マーケット管理者は利用者情報の送信を引き起こすアプリケーションの配布を防止することができる．これにより，マーケット自体の信頼性を担保でき，市場の活性化を図ることができる．公式マーケットでは既にマーケット上でのアプリケーション検査が実施されており[2]，マーケットにおけるアプリケーション検査の必要性は十分に認識されている．

一般的に，マーケットには多数のアプリケーション開発者からアプリケーションがアップロードされる．マーケット管理者は，アップロードされたアプリケーションを検査し，マーケット管理者が規定した安全基準（ポリシー）を満たすアプリケーションのみマーケットに登録する．ユーザは，マーケットに登録されたアプリケーションを自由にダウンロードすることができる．

アプリケーションの検査手法として，例えば，iOS 用の公式マーケットである App Store では，アプリケーションを数週間かけて静的に解析し，App Review Guidelines に従って開発されているかを厳密に審査している[7]．また，Android 用の公式マーケットである Google Play では，アプリケーション開発の自由度やアプリケーション登録の迅速性を重視しており，アップロードされたアプリケーションに対して必要最低限の検査 のみ実施している[8]．しかし，Google Play では，現在の検査で利用者情報の送信を引き起こすアプリケーションを十分に検知できておらず，安全面での改善が必要である．このため，本研究では，Android マーケット管理者がアップロードされたアプリケーションをポリシーに基づいて検査する手法を研究対象とする．具体的には，アプリケーションによる利用者情報の送信に関する情報を出力するアプリケーション解析手法を研究開発する．なお，ポリシーを満たす正常なアプリケーションも利用者情報を送信する場合がある．このため，ポリシーを満たすか判定する際の根拠情報となる，利用者情報の送信先や送信される情報も解析手法により特定する必要がある．

1.3 研究概要

マーケットにおけるアプリケーションを検査する手法として，以下の 3 つが挙げられる．

- アプリケーション開発者の身元確認
- アプリケーションの静的解析
- アプリケーションの動的解析

アプリケーション開発者の身元確認手法は，クレジットカード情報や身分証明書のコピーを開発者に提示させる手法である．しかし，この手法では，開発者の手間がかかるのみでなく，マーケット側の情報管理負荷が著しく高くなるため,アプリケーション登録の迅速性を重視する Android マーケットには適さない．

静的解析手法は，アップロードされるアプリケーションを動作させずに実行ファイルやソースコードの内容を解析する手法である．App Store が適用していると考えられる本手法では，アプリケーションが具備する不正な機能を特定できる一方で，自動化が困難であるため，検査のために長い時間と高い費用が必要となる．このため，この手法はアプリケーション登録の迅速性を重視する Android マーケットには適さないと考えられる．

動的解析手法は，アプリケーションの実行ファイルを動作させて，呼ばれた関数や変数の流れを解析する手法である．コードカバレッジを 100%にすることが困難であるという問題がある．ここで，コードカバレッジとは，プログラムコードに書かれた機能に対して動作を引き出せる範囲のことである[9]．一方で，動的解析手法では実際にアプリケーションをインストールした後に発生する可能性が高い動作を機械的に特定できる．このため，アプリケーションの効率性な検査に適していると考えられる．Google 公式マーケットでは，Bouncer と呼ばれる動的解析手法によるアプリケーション検査が行われている[2]．

以上から，本テーマでは，アプリケーションの検査手法として，アプリケーション数の増加に対応できる動的解析手法に着目した．特に，Android マルウェアの中でも，昨今社会問題となっている利用者情報を送信するアプリケーションの動的解析に着目した．

本研究では，利用者情報送信に関する既存の動的解析手法の問題を分析し，課題を明らかにするとともに，解決手法の検討を行った．さらに，解決手法の実装を行い，マーケットのアプリケーションを用いて検討内容を評価した．

2. 問題分析

本章では，本研究で対象とする従来の動的解析手法で実装されている解析機能の問題を分析した結果と，解決すべき技術的課題について説明する．

2.1 従来手法

現在有効な動的解析手法である Droidbox[10]/TaintDroid[11]では，テイント解析機能が実装されている．Droidbox は，TaintDroid による解析ログの統計処理機能や Java API トレース機能を追加した手法であり，テイント解析機能は TaintDroid と同等である．テイント解析は，情報にテイントと呼ばれる識別子を付与し，その情報が使用された場合は，使用された情報と共に伝搬するフローを解析する手法である．通常，テイント解析は，テイントが付与される情報群を示す source，テイントの伝搬を示す propagation，伝搬されたテイントを監視するポイントを示す sink という 3 つの構造で構成されている[12]．電話番号にテイントを付与して，ネットワークと外部ストレージに送信する場合における構造間の関連を図 1 に示す．

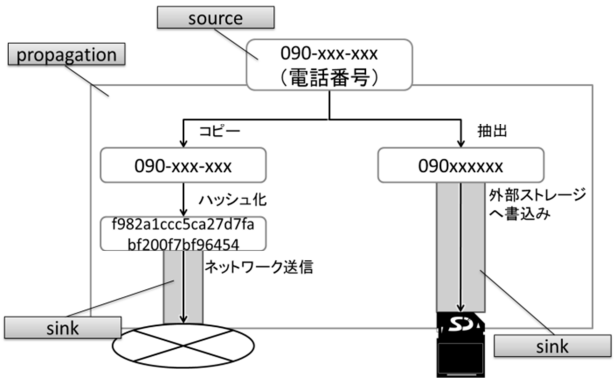

図 1　テイント解析構造間の関連

source は最初にテイントが付与される情報群である．TaintDroid においては，端末識別番号（IMEI）や電話番号といった利用者情報が source として扱われる．source は，32 ビットで重複がないように管理されている（図 2）．

propagation は，テイントが伝搬するフローである．特に，付与された情報群が，どのような操作が行われたときにそのテイントを伝搬させるかというルールを propagation policy という．TaintDroid においては，変数同士の単項演算やコピー，配列への挿入や値取得などが行われた際にテイントが伝搬する．

sink は source から伝搬してきたテイントを監視するポイントである．TaintDroid においては，ネットワーク送信，内部ストレージへの書込み，外部ストレージへの書込みが対象となっており，利用者情報送信の観点からは十分であると考えられる．

```
#define TAINT_CLEAR              ((u4)0x00000000)
#define TAINT_LOCATION           ((u4)0x00000001)
#define TAINT_CONTACTS           ((u4)0x00000002)
#define TAINT_MIC                ((u4)0x00000004)
#define TAINT_PHONE_NUMBER       ((u4)0x00000008)
#define TAINT_LOCATION_GPS       ((u4)0x00000010)
#define TAINT_LOCATION_NET       ((u4)0x00000020)
#define TAINT_LOCATION_LAST      ((u4)0x00000040)
#define TAINT_CAMERA             ((u4)0x00000080)
#define TAINT_ACCELEROMETER ((u4)0x00000100)
#define TAINT_SMS                ((u4)0x00000200)
#define TAINT_IMEI               ((u4)0x00000400)
#define TAINT_IMSI               ((u4)0x00000800)
#define TAINT_ICCID              ((u4)0x00001000)
#define TAINT_DEVICE_SN          ((u4)0x00002000)
#define TAINT_ACCOUNT            ((u4)0x00004000)
#define TAINT_HISTORY            ((u4)0x00008000)
```

図 2　TaintDroid における source 定義

2.2 従来手法の問題

TaintDroid では，sink の実装が十分である一方で，source と propagation に関して以下に示す問題がある．

source に関して，TaintDroid では，電話帳やセキュリティに関わる設定データやシステム設定データへのテイント付与が不十分であり，これらのデータの送信を検知できない．この問題を解決するためには，全ファイルや API やメモリーデータを source とする手法が考えられる．しかし，この手法では，source が必要とする仮想メモリおよび propagation が発生する機会が爆発的に増加し，メモリ使用量や CPU 処理性能に問題が発生する．このため，従来手法の問題を解決するためには，source の網羅性を効率的に高める必要がある．

propagation に関して，TaintDroid では，テイントが付与された情報群が，DB やファイルに外部出力された後に再度読み込ませた場合，テイントの伝搬が途切れてしまう．さらに，テイントが付与された情報群に依存してコードフローが決定される場合，テイントの伝搬が途切れてしまう問題が指摘されている[13]．この問題を解決するためには，正確な propagation policy を適用する必要がある．

source の網羅性欠如により利用者情報を送信するアプリケーションを特定できない問題は，現在の脅威に対抗できないという意味で早急に解決すべき問題である．このため，今回は，source の網羅性欠如を解決するために，source の範囲拡張および拡張された source に対する propagation 設計によって，利用者情報送信アプリケーション特定手法を検討した．

3. 提案手法

本章では，source 範囲拡張による利用者情報送信アプリケーション特定手法について，設計と実装を行った結果を報告する．

3.1 設計

TaintDroid は，DB に格納された利用者情報へのテイント

が不十分であった．しかし，DB に格納された利用者情報の送信が，大きなインシデントの原因となる可能性は非常に高いと考えられる．特に，アドレス帳に相当するコンタクトリストや, 通話履歴およびアクセス先 URL 履歴などに代表される，DB に格納された利用者情報の送信は，当該端末がアクセスする先の情報を含んでおり，影響範囲が広い．

(1) **DB へのテイントにおける source**

本手法では，DB に格納された情報に着目してテイントを付与する．DB を利用する source と利用しない source は表 1 の通りである．

表 1　DB を利用する source と利用しない source

DB を利用する source	DB を利用する source
端末識別番号（IMEI）	コンタクトリスト
契約者識別番号（IMSI）	通話履歴
SIM 識別番号（ICCID）	URL 履歴
自端末電話番号	ブックマーク情報
位置情報	セキュア設定情報
GPS 位置情報	システム設定情報
Wifi 位置情報	
直前位置情報	
カメラ画像	
SMS 情報	
マイク入力データ	

DB へのテイント付与手法は，以下の 2 種類に分けられる．

- DB 全体へテイントを付与する手法
- DB の各レコードへ付与する手法

ここで，DB 数を N_{db}，全てのレコード数を N_{rc}，テイントタグ容量を T とすると，それぞれの場合に必要なテイント保持容量 S_a，S_b は以下の通り計算できる．

$$S_a = N_{db}T$$
$$S_b = N_{rc}T$$

以下では，一般的に $N_{db}<N_{rc}$ であると仮定する．S_a は S_b に比べて，テイント保持に必要な容量が少ないことがわかる．しかし，DB 全体へテイント付与する手法は，正確に propagation が行われない場合がある．例えば，テイントが付与された値が DB の既存レコードに上書きされると，DB のテイントが更新される．しかし，上書きされたレコードとは別のレコードから情報を取得した場合，本来 propagation されるべきではない情報がテイント付与されてしまう．一方で，DB の各レコードへテイントを付与する場合は，各レコードの propagation が正確に行われる．しかし，DB の各レコードへのテイント付与は，テイントを保持するための容量が大きくなるという問題点がある．

そこで本手法では，テイントの状態を保持するための DB を新たに作成し，初期状態から変更されたレコードの状態を記録する. この DB は, 変更されたレコード情報と，変更後のテイントを組みで保持する．初期状態から変更されていないレコードを取得する場合，あらかじめ DB の種類ごとに定められたテイントを付与する．操作されたレコード数を N_{op} とすると，本手法におけるテイント保持容量 S_c は以下の通り計算できる．

$$S_c = N_{op}T \quad (0 \le N_{op} \le N_{rc})$$

以上から，S_a，S_b，S_c における大小関係は以下の通りである．

$$S_c \le S_a \le S_b \quad (N_{op} = 0)$$
$$S_a \le S_c \le S_b \quad (1 \le N_{op} \le N_{rc})$$

よって，本手法におけるテイント保持容量は，レコード全てにテイント付加を行う場合と比べて，少ないことがいえる．以上から，利用者情報の送信を特定するために必要な source の網羅性を効率的に改善できると考えられる．

(2) **DB へのテイントにおける propagation policy**

テイントの状態を保持するための DB は，アプリケーションが DB のレコードを，挿入，更新，削除されたときに状態を更新する．挿入及び更新する情報のテイントを T_a，レコードに記録されたテイントを T_b，操作後のテイントを T_c とすると，それぞれの操作における propagation policy を以下に示す．

- 挿入
  $T_c = T_a$
  挿入する情報のテイントを引き継いで propagation を行う．
- 更新
  $T_c = T_a \cup T_b$
  更新される前の情報と, 挿入する情報のテイントを引き継いで propagation を行う．
- 削除
  $T_c = 0$
  テイントの値を 0 とする．これは，propagation されなくなることを示す．

3.2 実装

3.1 に示した設計を実現するために, TaintDroid に対して，テイントの状態を保持するための DB を介した DB テイント付与機能追加を行った. TaintDroid は Android Open Source Project (AOSP)の標準 OS ソースコードを基に実装されているため，今回の source 範囲拡張も同様のソースコードに基づいて実装した．なお，TaintDroid では，source のテイント付与定義と propagation policy および sink 定義を，仮想 VM (dalvik VM)と Java フレームワークライブラリ内で実装している．

Android において，DB にアクセスするためには，ContentProvider を利用する必要がある. ContentProvider は，アプリケーションから発行された SQL 文を引数として，

DB からレコードを取得するクラスである．引数として，対象の DB を指定する URI（Uniform Resource Identifier）部と，DB から取得する条件を指定するクエリ部が存在する．

テイントの状態を保持する DB を実現するため，ContentProvider における DB の挿入，更新，削除の関数を改良した．アプリケーションが DB のレコードを取得する場合，改良された関数を経由して初期状態から変更されたテイントを取得することができる．また，初期状態から変更されていないレコードを取得する場合，あらかじめ DB の種類ごとに定められたテイントを付与する．このとき，Android の DB にアクセスするための URI を利用する．この URI は，管理される情報によってパスが定められており，URI ごとに source を割り当てることができる．URI と source の対応関係を表 2 に示す．

表 2　URI と source の対応

URI	source
content://com.android.contacts	コンタクトリスト
content://call_log	通話履歴
content://blowser/bookmarks	URL 履歴， ブックマーク情報
content://settings/secure	セキュア設定情報
content://settings/system	システム設定情報

4. 実験と評価

本章では，source 範囲拡張および propagation 改良を行った解析用 OS を用いて，実マーケット上のアプリケーションを検査することで，提案手法の評価を行った結果を報告する．

4.1 実験

本実験では，解析用 OS を用いて，利用者情報送信を引き起こす検体数の評価および利用者情報送信宛先数の評価を行うことを目的とする．

また，機械的に収集可能である非公式マーケットから収集したアプリケーションを検体とした．提案手法の統計結果は，アドウェアやマルウェアおよび通常のアプリケーションによって分布が変化する可能性がある．このため，収集したアプリケーションをアンチウイルスソフトウェア（ESET NOD32）で分類する事前実験を行った．具体的には，アドウェアと判定される検体と，マルウェアと判定される検体（アドウェアは含まない）およびアドウェアやマルウェアと判定されなかった検体群に分類した．今回は，各検体群から各々任意の 300 検体を抽出し，合計 900 検体を用いて実験を行った．今回の実験では，AOSP バージョン 2.1 をベースとした OS で解析を行った．

次に，実験の流れと出力される情報について述べる．まず，解析用 OS を起動し，テイントに関わるログを監視する．次に，解析対象の検体をインストールする．検体がインストールされた 30 秒後に，ログを出力して解析用 OS を終了する．ログには，以下の情報が出力される．

- sink に送信された source の種類
- sink に送信されたテイント付与データ
- sink に送信されたタイムスタンプ
- sink 情報（source を送信した宛先 IP アドレスまたはストレージ書込みを行った宛先パス名）

4.2 利用者情報送信を引き起こす検体数の評価

利用者情報送信を引き起こす検体数の評価を行うため，sink 情報が出力された検体について検討する．既存手法で sink 情報が出力された検体数 N_{cur}，本手法で sink 情報が出力された検体数 N_{ext}，本手法で新たに sink 情報が出力された検体数 N_{dif} を集計した結果を表 3 に示す．N_{cur} としては，全体の検体のうち 244 検体の存在が確認できた．この中でも，アドウェアとマルウェアはそれぞれ 115 検体，104 検体と数が多いことが確認できた．次に，N_{ext} としては，全体の検体のうち 263 検体の存在が確認でき，N_{cur} と比較して 19 検体の増加が確認できた．この 19 検体のうち，アドウェアであったものが 8 検体，アドウェアやマルウェアと判定されなかった検体群が 11 検体であった．以上から，本手法は既存手法では確認できなかった利用者情報送信アプリケーションを新たに特定できることが得られた．ただし，アドウェアでもマルウェアでもないが sink に送信する検体は，誤検知の可能性があるため，5 章で考察を述べる．

表 3　本手法で sink 情報を確認できた検体数

	N_{cur}	N_{ext}	N_{dif}
アドウェア	115	123	8
マルウェア	104	104	0
上記以外	25	36	11
全体	244	263	19

4.3 利用者情報送信を宛先数の評価

利用者情報送信宛先数の評価するため，sink 情報で得られた sink 数について検討する．既存手法で確認できた sink 数 I_{cur}，本手法で確認できた sink 数 I_{ext}，本手法で新たに確認できた sink 数 I_{dif} を集計した結果を表 4 に示す．I_{cur} としては，全体の検体のうち 587 件の存在が確認できた．一方 I_{ext} としては，全体の検体のうち 746 件の存在が確認できた．I_{cur} と比較すると，全体で 168 件，IP アドレスは 97 件，ストレージパスは 71 件の増加が確認できた．

以上から，本手法はアドウェアに対して既存で確認できなかった sink 情報が取得できることが明らかになった．特に，sink 情報のうち，IP アドレス情報に関しては，従来の source のみを用いた結果に追加情報として付加される場合が多数確認できた．IP アドレス情報は，悪性通信の検知に活用可能であるため，提案手法によって有効な情報を抽出

できたと考えられる．ただし，検体によっては送信先が分散する場合が確認できたため，5 章で考察を述べる．

表 4　本手法で得られた sink 数

	I_{cur}	I_{ext}	I_{dif}
IP アドレス数	395	492	97
ストレージパス数	183	254	71
全体	578	746	168

5. 考察

アドウェアでもマルウェアでもないが sink に送信する検体については，4.2 節で示した通り 36 検体が確認できた．このうち，33 検体が広告モジュールが実装され，利用者情報を外部に送信していた．広告モジュールを実装した検体は，IMEI，電話番号情報，位置情報，セキュア設定情報のいずれかもしくは複数を送信していることが確認できた．また，その他の 3 検体は，広告モジュールは実装されておらず，アプリケーションフォルダのみに書き込みを行うものであった．そのため，利用者情報を送信するアプリケーションであるとはいえない．以上から，sink 範囲を改良する必要があることが得られた．

送信先が分散する検体については，既存手法では 3 つの IP アドレス，本手法では 8 つの IP アドレスを sink 情報とする検体が確認できた（表 5）．IP アドレス D,E,F,G,H は本手法で新たに得られた IP アドレスである．さらに，IP アドレス A に対して拡張した source 情報を送信することも確認できた．ただし，静的解析を行った結果，14 の広告モジュールを実装していたことが確認できたが，本手法では全ての広告モジュールは確認できなかった．これは，ユーザインタラクションを必要とする動作や，時間に依存した動作を実行できていないため，コードカバレッジが満たされていないことを意味する．根本的にコードカバレッジを高める方法は今後の課題である．

表 5　IP アドレスと送信する source の対応

IP アドレス	送信する source
IP アドレス A	ICCID，セキュア設定情報
IP アドレス B	IMEI, IMSI，電話番号
IP アドレス C	IMEI, IMSI，電話番号
IP アドレス D	セキュア設定情報
IP アドレス E	セキュア設定情報
IP アドレス F	セキュア設定情報
IP アドレス G	セキュア設定情報
IP アドレス H	セキュア設定情報

6. まとめと今後

本研究では，利用者情報の送信を引き起こす Android アプリケーションを特定する技術として，従来手法では存在しなかった，DB 情報の送信を特定する手法の提案と評価を行った．

提案手法では，現在までの Android マルウェアにおいて主流であった DB を利用しない情報の送信以上に，DB を利用する情報の送信が重大なインシデントを引き起こす可能性が高い点に着目し，テイント解析手法の拡張するための手法を示した．また，提案手法を実装してマーケットから収集したアプリケーションを検査した結果，提案手法でのみ特定可能な情報が存在することが明らかになった．

今後は，sink 範囲の改良やコードカバレッジの問題を検討するために，アプリケーション収集を継続的に行って問題分析を実施し，提案手法の改善を図る．

参考文献

[1] トレンドマイクロ，インターネット脅威レポート 2011 年度
http://jp.trendmicro.com/jp/threat/security_news/monthlyreport/article/20120106083242.html
[2] Google Mobile Blog, Android and Security
http://googlemobile.blogspot.jp/2012/02/android-and-security.html
[3] Symantec，日本の Android ユーザーから利用者情報を盗み出す”The Movie”マルウェア
http://googlemobile.blogspot.jp/2012/02/android-and-security.html
[4] 産経ニュース, スマホアプリで 76 万件分の利用者情報流出か「全国電話帳」インストールに注意
http://sankei.jp.msn.com/affairs/news/121006/crm12100617380008-n1.htm
[5] ZDNet Japan，なぜ IMEI の利用が問題視されたのか--利用者情報とプライバシーをめぐって
http://japan.zdnet.com/security/analysis/35009620/
[6] Symantec，Android.Expresspam の作成者グループ，Gcogle Play から「ANDROID EXPRESS の PLAY」にリニューアル
http://www.symantec.com/connect/blogs/androidexprespam-gcogle-play-android-express-play
[7] App Developer, App Review Guidelines
https://developer.apple.com/appStore/guidelines.html
[8] Google, Google Play デベロッパープログラムポリシー
http://play.google.com/about/developer-content-policy.html
[9] Bullseye testing Technology, CodeCoverage Analysis
http://www.bullseye.com/coverage.html
[10] The Honeynet Project, droidbox
http://code.google.com/p/droidbox/
[11] William Enck et al., “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones”, Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2010.
[12] James Clause et al., “Dytan: A Generic Dynamic Taint Analysis Framework”, Proceedings of the International Symposium on Software Testing and Analysis(ISSTA), 2007.
[13] L.Cavallaro et al., “On the Limits of Information Flow Techniques for Malware Analysis and Containment”, Proceeding of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment(DIMVA), 2008.