# Androidアプリがサードパーティに送信する情報の調査: JDIを用いた動的解析機構

程 斌[1]　細谷 竜平[1]　鈴木 嵩人[2]　齋藤 孝道[2]　森 達哉[3]

**概要**：モバイルアプリにおいては，様々な理由により，端末内の情報がサードパーティに送信されることがあることが知られている．しかしながら，どのような情報が送信されているのかを把握することは困難であること多い．そこで，本論文では Java Debug Interface（JDI）を用いて，モバイルアプリ実行中にどのような情報が送信されているのかを調べる動的解析機構を実装した．そして，その解析機構を用いて Google Play ストアでリリースされている人気アプリの動作を解析し，Android のアプリがどのような情報をどのように外部に送信しているのかについて調査を行った．

**キーワード**：モバイル，Android，動的解析，プライバシー

# Measurement Study of Information Transmitted to Third-party by Android App: Dynamic Analysis System with JDI

Bin Cheng[1]　Ryohei Hosoya[1]　Takahito Suzuki[2]　Takamichi Saito[2]　Tatsuya Mori[3]

***Abstract:*** In the mobile apps, it is known that information in the device is transmitted to a third-party for various reasons. However, it is often difficult to grasp what kind of information is being transmitted. Therefore, in this paper, we implemented a dynamic analysis system that uses Java Debug Interface (JDI) to investigate what kind of information is being transmitted while using mobile apps. Then, using that system, we analyzed the behavior of popular apps released on the Google Play Store and investigated what kind of information Android apps are transmitting to the third-party.

***Keywords:*** Mobile, Android, Dynamic Analysis, Privacy

## 1. はじめに

モバイル端末は発売当初から国内外で普及してきている．2018 年現在では，世界中の主要な国でのモバイル端末の使用率は 50 %を超えていると言われている．さらに日本では，使用率が 64 %以上となっている [9]．また，30 代以下の年齢層で，スマートフォンの利用率は 83 %も超えている [10]．

スマートフォンはアプリケーション，いわゆるモバイルアプリの利用が必須である．モバイルアプリにおいては，様々な理由により，端末内の情報が取得され，サードパーティに送信されることがあることが知られている．そんな中，モバイルアプリの利用者に関わる情報が意図せずに送信される可能性も考えられる．

元来，モバイルアプリでは，OS が定めた API を利用することにより，利用者及び端末に関する情報を取得し，送信することができる．しかしながら，アプリが送信する情報の詳細が開示されないケースが多いため，利用者は実際に送信されている情報を把握できない．このような送信状況を調査するためには，モバイルアプリを個別に解析し，その挙動を分析する必要がある．

1 明治大学大学院
Graduate School of Meiji University
2 明治大学
Meiji Univevrsity
3 早稲田大学
Waseda University

モバイルアプリの解析は，主に静的解析と動的解析に分けられるが，いずれにも一長一短がある．静的解析は，モバイルアプリのソースコードを抽出し，その挙動を予測する方法である．しかし，実際の利用において実行されないコードが含まれている場合もあるため，解析という観点では正確性に欠ける可能性がある．それに対して，動的解析では実際にアプリを操作し，その挙動を解析する手法である．動的解析の場合，モバイルアプリの実際の挙動をもとに解析を行うので，実行された機能であれば確実な解析結果を得られると言える．しかしながら，静的解析ほど解析は容易ではない．

以前の我々の研究 [12] において，Android Studio のデバッガ機能を利用して Android アプリに対する動的解析を行った．しかし，手動での解析のため，効率の面には改善の余地があった．

そこで，本論文では Java Debug Interface（JDI）[6] を用いて，Android アプリ実行時に取得，送信されている情報を自動的に調査する動的解析機構を作成した．

そして，その動的解析機構を用いて Google Play ストアでリリースされている人気アプリの動作を解析し，Android のアプリがどのような情報をどのようにサードパーティに送信しているのか調査を行った．

調査の結果，調査対象とした 305 個の Android アプリ全てが端末から利用者情報を取得していることがわかった．そして，196 個はサードパーティへ利用者情報を送信していることがわかった．その中でも，特に以下のようなことがわかった．

- ADID を送信しているアプリの数は全体の 45.6%を占めている．
- Android ID を送信しているアプリの数は 28.5%を占めている．
- Wi-Fi に関する情報を送信しているアプリの数は 11.8%を占めている．

## 2. 関連研究

### 2.1 モバイルアプリが取得している利用者情報に関する研究

モバイルアプリが取得している利用者情報に関する研究は，マーケット立ち上げ当時から現在に至るまで盛んにされている．

細谷ら [11] は，コード解析を利用してアプリが取得している情報を API レベルで分析した．そして，国内でリリースされている 1,704 個のアプリに対し，コード解析を行った．その結果，40.08%は端末のシリアル番号を，71.06%は位置情報を，39.26%はインストール済みアプリケーション名を取得していることを示した．

福田ら [13] は JDWP と呼ばれる Android アプリにおけるデバッグ情報のやり取りを行うプロトコルを用いて，Android アプリに組み込まれている外部モジュールが外部へ送信しているグローバル ID を調査した．その結果，国内でリリースされている Android アプリの 11%が IMEI や IMSI といった，Android アプリにおいて非推奨とされているグローバル ID を送信していたことを示した．

Grace ら [3] は，広告ライブラリに関連するプライバシーリスクを特定する静的分析ツールである AdRisk を提案した．このツールを用いて Google Play ストアでリリースされている 100,000 個のアプリケーションを調査し，52,067 個のアプリケーションが広告ライブラリを使用していることを示した．また，その 31 %が複数の広告ライブラリを使用していることも示した．さらに，彼らは調査した 100 の広告ライブラリの大部分が利用者の端末情報を収集していることを示した．

Seneviratne ら [7] は，無料のアプリケーションが利用者の情報を取得していると言われているが，同様に，有料のアプリケーションも利用者の情報を収集していることを示した．有料アプリケーションの 60%が利用者の情報を収集したのに対し，無料アプリケーションでは 85%であった．また，彼らは収集された 3,605 個の Android アプリケーションの 20%が 3 つ以上の広告ライブラリやアナリティクスライブラリに接続されていることを示した．

### 2.2 モバイルアプリが取得している情報を用いた端末識別に関する研究

モバイルアプリが取得している利用者情報に関する研究の一方で，モバイルアプリが取得している情報を用いた端末の識別に関する研究も行われている．

Kurtz ら [5] は，自作のモバイルアプリをインストールさせ，そのモバイルアプリから収集した情報を用いて端末のトラッキングを行い，100%の端末が識別可能，97%の端末が時間経過を伴う識別，つまりトラッキングが可能であることを示した．Kurtz らは，ハードウェアに関する情報など，様々な情報を収集したが，特にインストール済みアプリがトラッキングに十分な情報量を持っていることを示している．

Wu [8] らは，Android アプリから採取可能であり，なおかつ利用者によるパーミッションの必要がない特徴点 38 個を用いて端末の識別を行った．実験では 2,239 端末から採取した 50,830 個のサンプルを対象に調査し，その結果，Precision が 99%以上，Recall が 98%以上となった．

Han ら [4] は，被験者 20 名の端末に対して 3 週間の測定を行い，現実世界におけるモバイルアプリによるサードパーティトラッキングに関する研究を行った．その結果，アプリに紐付ける Web サイトのうち 36%（1824 件中 655 件）が利用者を追跡していることを明らかにした．その中の 37%はハードウェア識別子（主に Android ID と IMEI）を利用して追跡を行っていた．また，広告と解析サービス

のライブラリが多くのモバイルアプリの中に埋め込まれ，それらによる重度なトラッキングが行われていることがわかった．

## 3. 関連知識

### 3.1 JPDA

Java Platform Debugger Architecture（JPDA）とは，仮想マシンをデバッグするための一連のツールとインタフェースである．

Java プログラムはすべて Java 仮想マシン（JVM）上，Android アプリは Dalvik 仮想マシン（Dalvik VM）上で実行されている．実際には，Java プログラムや Android アプリをデバッグする際，ターゲット仮想マシン（JVM または Dalvik VM）から現在の実行状態を要求したり，特定の命令を仮想マシンに発行したりするなどの必要がある．JPDA を利用することにより，これらの機能を実現することができる．

図 1 に JPDA の概念図を示す．JPDA は，独立した 3 つの階層で構成されている．JPDA の階層は，Java Virtual Machine Tool Interface（JVMTI）（3.2 節参照），Java Debug Wire Protocol（JDWP）（3.3 節参照），および Java Debug Interface（JDI）（3.4 節参照）とそれぞれ呼ばれる．

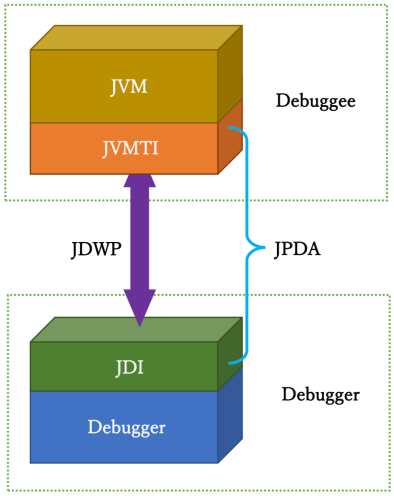


図 1　JPDA の概念図

### 3.2 JVMTI

Java Virtual Machine Tool Interface（JVMTI）とは，JVM が提供している一連のインターフェースである．JVMTI は JPDA の 3 層構造の最も低い階層であり，直接 JVM に対したデバッグ機能を提供する．

### 3.3 JDWP

Java Debug Wire Protocol（JDWP）とは，JPDA の 3 層構造の中間に位置しており，Java プログラムのデバッグのために設計された通信プロトコルである．JDWP では，デバッガとデバッグの対象（デバッギ）の間で送信されるデータ（デバッグ情報やデバッグコマンドなど）の形式が定義されている．JDI（3.4 節参照）ベースのデバッガがターゲット仮想マシンとデバッグ通信を行う際には，JDWP を使う必要がある．

### 3.4 JDI

Java Debug Interface（JDI）とは，JPDA の 3 層構造の最上位にあたるインタフェースで，デバッガが必要とするいくつかのデバッグインタフェースを備えている．これらのインタフェースに基づいて，デバッガは，ターゲット仮想マシン上でロードされたクラスや実行中のインスタンスといった，ターゲット仮想マシンの実行状態の取得ができる．さらに，デバッガは，ターゲット仮想マシンのスレッドの一時停止と復元，ブレークポイントの設定など，ターゲット仮想マシンの実行を制御することもできる．つまり，JDI を用いて，独自のデバッグツールを開発することができる． デバッガの開発者は，これらの JDI によって提供されるインタフェースを使用して，Java のデバッガをカスタムし，特定のニーズに対して拡張することにより，特殊なデバッグツールを開発できる．

### 3.5 ADB

Android Debug Bridge（ADB）は Android 端末とデバッグ情報のやり取りを行うツールである．Android 上のアプリは全て Dalvik VM の中で実行される．Dalvik VM は JVM とは異なる仮想マシンである．Dalvik VM は JVMTI の機能を提供していないが，JDWP の機能が含まれている．そのため，JPDA を介して Dalvik VM に対するデバッグを行う時，JVMTI を使えず，代わりに ADB を利用する必要がある．Android アプリのデバック時に，ADB は JDWP に基づいた通信路を生成し，JVM と同じように，JPDA を介して Dalvik VM とのデバッグコマンドの通信などを実現できる．

図 2 に ADB を用いた Dalvik VM に対するデバッグのイメージを示す．デバッガは実行中のアプリのスタックフレーム情報やインスタンスフィールド情報やメソッドの実行情報などを取得することができる．

### 3.6 Android API

Android API は Android OS が定義している一連のインターフェイスであり，これらのインターフェイスを利用して，Android アプリは，様々な OS 機能を使用することができる．ほとんどの Android アプリは Android API を利用して開発されている．Android アプリによっては，Android API を利用して，利用者及び端末に関する情報を取得したり，通信を行ったりするものなど，多数存在する．よって，Android API を観測することは，Android アプリ

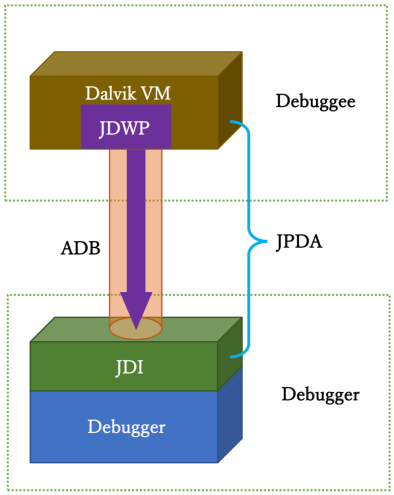


図 2　ADB 用いた Dalvik VM に対するデバッグのイメージ

の挙動を解析するには有用な手段である．

## 4.　動的解析機構（AADA）

本節では，本研究において実装した動的解析機構，Android Application Dynamic Analyzer（以降，AADA と呼ぶ）について解説する．

### 4.1　AADA の概要

AADA は，JDI をベースとして実装された，JPDA のデバッグシステムの性質を利用する Java のデバッガである．JPDA では前述したように，ターゲット仮想マシンの実行状態をリアルタイムで取得できる．それによって，AADA は Dalvik VM 上で実行している Android アプリにおいて，API の呼び出しの観測を行う．これにより，アプリの挙動を解析することができる．

本論文では，アプリ内で API 呼び出しを観測することで，利用者情報が端末内で取得されているか，またサードパーティに送信されているかを判定する．

### 4.2　AADA の構成

本節では，AADA の構成について解説する．AADA の構成図を図 3 に示す．

AADA を構成する各種機能について，1 つずつ解説する．

**mainController**

mainController は，AADA の全体の機能をコントロールする役割を担う．具体的には，mainController の中では，解析のルール（どのような挙動で観測を行うか）をコントロールする ruleController 機能，および解析結果から有用な内容を抽出し，それをログファイルとして出力する output 機能を持つ．ruleController 機能および output 機能の内容を変更することで，状況に応じて，解析方式および出力データの形式を変更することができる．

**JVMConnector**

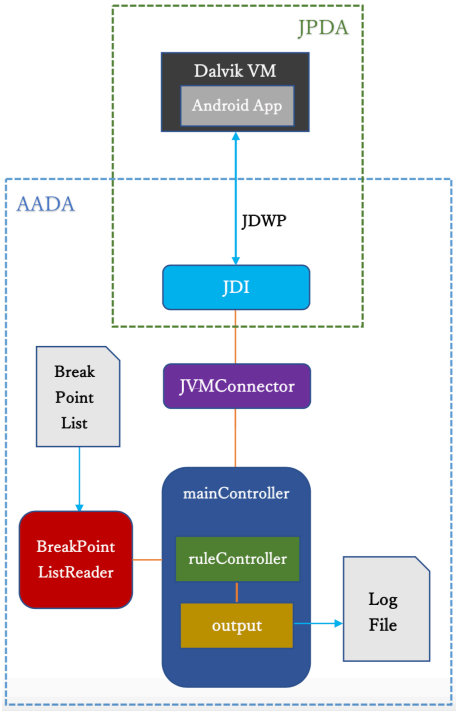


図 3　AADA の構成図

JVMConnector は JDI が提供しているコネクターを利用してターゲット仮想マシンへの接続を行う機能を持つ．

**JDI**

JDI は，3.4 節で述べたように，JPDA を通して，ターゲット仮想マシンに対するデバッグ操作を AADA に行わせる一連のインターフェイスである．

**BreakPointListReader**

BreakPointListReader は，後の 4.3 節で解説する API 観測に使用される Break Point List ファイルを読み込んで，mainController にブレークポイントの情報を伝達する機能を持つ．

### 4.3　AADA の動作

本節では，AADA の動作について解説する．AADA は解析対象のアプリに対して，以下のような手順で解析を行う．

(1) 解析対象アプリの実行

AADA を起動する前に，解析対象の Android アプリを起動する．Android アプリが実行された後，ADB を利用して Android アプリのプロセスに対して JDWP の通信路を確保する．

(2) **AADA の起動・Dalvik VM との接続**

AADA を起動する．そして，JVMConnector が JDWP 経由で Dalvik VM に対する接続を試行する．接続が成功すると，Android アプリに対する解析作業が自動的に開始される．

(3) ブレークポイントの読み込み

API のブレークポイントを読み込み，観測対象となる API の呼び出しを特定する．BreakPointListReader は Break Point List ファイルを読み込み，このリストファイルに存在する各項目（観測対象となる API のクラス，メソッド名となる）に対して BreakPoint オブジェクトを生成する．BreakPoint オブジェクトには観測対象となる API のクラス名，メソッド名および有効化状態の 3 つの変数を持つ．生成された BreakPoint オブジェクトは，全て mainController の中に保存される．

(4) **ブレークポイントの設置**

mainController が保存されている BreakPoint オブジェクトに基づいてブレークポイントの設置作業を行う．具体的には，mainController は Android 上の Dalvik VM へ，各 BreakPoint オブジェクトに保存している API のクラス名とメソッド名を送信し，ブレークポイントの設置するイベントの発行および有効化を要求する．これにより，Android アプリ実行中に毎回ブレークポイントの到達が検知される際，観測対象となる API が呼び出されることがわかる．

(5) **API の呼び出し状況の取得**

観測対象となる API の呼び出しを検知される度，mainController は ruleController で事前に設定したルールで，API の状態を取得する．本論文では，API が呼び出された瞬間のスレッドの状況，スタックの中の呼び出し関係，呼び出した API のメソッドの引数およびローカル変数を取得する．

また，観測対象の API の状態が実行中に変化する可能性があるので，呼び出される瞬間の状態だけではなく，API の戻り検知ポイントを設置し（(6) で説明する），戻り検知も行う．API の戻りが検知される際には，その戻った直前，もう一度その瞬間のローカル変数の内容を取得する．そして API の戻り値も取得する．これによって，解析でより多くの情報を取得することができる．

戻り値を取得した後，この 1 回分の API 呼び出しに対しての状態取得が完了するため，この 1 回分の戻り検知ポイントを削除する．

(6) **戻り検知ポイントの設置**

観測対象となる API の戻りを検知するため，戻り検知ポイントを設置する．mainController は毎回ブレークポイントの到達が検知される際，この呼び出した API のメソッドの位置を一つのリストに記録する．このリスト（以下，API 呼び出しリストと呼ぶ）にはすべての観測対象の API の毎回の呼び出しを順に記録する．また，API 呼び出しリストに存在する各 API に対してメソッドの戻りを検知するイベントの発行を Darvik VM に要求する．

(7) **実行結果の出力**

（5）で取得した動的解析の結果は，mainController の output 機能によって，ログとして出力される．解析結果には対象の API が呼び出された時点，および戻る時点それぞれの時刻，スレッドのスタックフレーム情報，引数，ローカル変数の内容，戻り値を含む．これらの情報を抽出しやすいフォーマットでログファイルを生成する．

## 5. 調査方法

本論文では，4 節にて説明した AADA を用いて，Google Play ストアでリリースされている各カテゴリの人気アプリに対して動的解析を行う．

### 5.1 調査対象のアプリ

本論文では，Google Play ストアの各カテゴリ別 (Google cast, Wear OS アプリを除いた 34 カテゴリ) 無料アプリダウンロード数ランキング（2018 年 7 月時点）上位 10 位の中，重複したものを除く，計 305 個のアプリを調査対象とした．

### 5.2 調査対象の利用者情報

本論文で調査対象とする利用者情報については端末，ネットワークおよび個人の特定の可能性がある以下のような情報に着目する．

**Android ID**

Android 端末を識別するための識別子である，ファクトリーリセットによって変更できる．

**Advertising ID（ADID）**

広告に用いられるグローバル ID である，利用者によってリセットすることができる．

**International Mobile Equipment Identity（IMEI）**

携帯電話に紐づく一意な識別子である，基本的に変更できない．

**Universally Unique Identifier（UUID）**

アプリを一意に識別するために用いられるグローバル ID である，アプリの再インストールによって変更できる．

**Globally Unique Identifier（GUID）**

UUID の実装の一つで，アプリの再インストールによって変更できる．

**Wi-Fi に関する情報**

端末が WiFi 機能に対応しているかどうかや端末が接続している WiFi アクセスポイントの名前である Service Set Identifier（SSID）などの情報を含む．

**GPS 情報**

端末の GPS センサーが取得した位置情報である．

**利用者の連絡先**

表 1　情報を取得する API および通信用の API

| 情報を取得する API の種類 | Android API 名 |
|---|---|
| Android ID | android.provider.Settings$Secure.getString |
| UUID | java.util.UUID.toString |
| ADID | com.google.android.gms.ads.identifier.AdvertisingIdClient$Info.getId |
| IMEI | android.telephony.TelephonyManager.getDeviceId |
| GPS 情報 | android.location.LocationManager.getGpsStatus |
| 電話番号 | android.telephony.TelephonyManager.getLine1Number |
| **通信用 API の種類** | **Android API 名** |
| 通信内容 | libcore.io.Posix.sendto |
| | com.android.org.conscrypt.OpenSSLSocketImpl$SSLOutputStream.write |
| HTTP リクエストの内容 | com.android.okhttp.internal.huc.HttpURLConnectionImpl.connect |
| | org.apache.http.impl.client.DefaultRequestDirector.execute |
| 通信先の URL | android.webkit.WebView.loadUrl |
| | android.webkit.WebView.postUrl |
| | android.webkit.WebView.loadDataWithBaseURL |
| ブラウザ App に送った URL | android.app.Activity.startActivityForResult |
| | android.app.Activity.startActivityAsUse |

メールアドレス，電話番号などの情報を含む．

**利用者本人に関する情報**

性別，誕生日などの個人情報を含む．

### 5.3　調査対象の Android API

本論文で調査対象とした Android API を表 1 に示す．調査対象の Android API は，具体的に 2 種類に分かれている．1 つは利用者及び端末に関する情報を取得する API であり，もう 1 つは通信を行う API である．

### 5.4　Android アプリに対する調査方法

本論文では，1 つの解析対象アプリに対し，以下のように調査を行う．

(1) **AADA による前処理**

AADA を実行し，Dalvik VM との接続を確立し，解析対象アプリに対するブレークポイントの読み込みなどを行う．

(2) **解析対象アプリの操作**

AADA の環境下で実際に解析対象アプリを操作する．このとき，可能な限りすべてのアプリ内のページにアクセスし，なおかつ 10 分以上の操作を行うようにする．

(3) **解析結果のフィルタリング**

AADA による解析結果のログファイルに対してフィルタリングを行う．今回は，解析対象の利用者情報が取得および送信されているかを特定するため，それらに関するキーワードを選定し，そのキーワードが含まれている行をフィルタリングする．選定したキーワードの一部を表 2 に示す．また，API の呼び出し元および情報の送信先をフィルタリングによって特定することで情報送信の相手がサードパーティであるかどうかを判定する．

表 2　解析結果のフィルタリングに使用したキーワードの一部

| 利用者情報の種類 | キーワード |
|---|---|
| Android ID | ”android_id”，”android id”<br>端末に設定している Android ID |
| ADID | ”ad_id”，”ad id”，”advertisingid” |
| IMEI | ”imei”，端末に設定している IMEI |
| UUID | ”uuid” |
| GUID | ”guid” |
| Wi-Fi に関する情報 | ”wi-fi”，”ssid”<br>接続した Wi-Fi の SSID |
| GPS 情報 | ”gps” |
| メールアドレス | ”mail”，事前に設定したメールアドレス |
| 電話番号 | ”phone number”，”phone_number” |
| 性別 | ”gender” |
| 誕生日 | ”birthday”，事前に設定した誕生日 |

### 5.5　調査方針

本論文では Android アプリの挙動を把握するため，以下のポイントに絞って解析を行う：

- **観測対象 API の呼び出しの有無**
  所定の Android API にブレークポイントを設置することで API の呼び出しを確認する．これにより，利用者及び端末に関する情報が取得されることや外部との通信が始まることがわかる．
- **呼び出した API が属するスレッド情報の取得**
  呼び出した API が属するスレッドの情報を取得することによって，スタックフレームのリストを抽出できる．これにより，API の呼び出し元を特定でき，情報の取得行為及び通信行為を行う主体がわかる．
- **呼び出した API のスタックフレーム情報**
  スタックフレーム情報を取得することで，呼び出した API の引数，ローカル変数の内容を取得することができる．これにより，取得される情報の内容及び送信される内容がわかる．
- **呼び出した API の戻り値**
  API 呼び出しから戻る際，戻り値を取ることができる．

これにより，一部の API が取得した情報の内容がわかる．また，情報を取得する API に対して，その呼び出しを中心として観測し，通信用 API に対して，その通信の内容を中心として観測する．

## 6. 調査結果

### 6.1 全体結果

調査の結果，今回調査した 305 個の人気 Android アプリ全てが，端末から利用者情報を取得していることが確認できた．その中でも，195 個（全体の 63.9%）はサードパーティへ利用者情報を送信していることが確認できた．

各種類の利用者情報を取得・（サードパーティへ）送信している Android アプリの数と割合を表 3 に示す．

**表 3** アプリによる利用者情報の取得・送信状況

| 利用者情報の種類 | 取得するアプリ数 | 送信するアプリ数 |
|---|---|---|
| Android ID | 141 (46.2%) | 87 (28.5%) |
| ADID | 163 (53.4%) | 139 (45.6%) |
| IMEI | 33 (10.8%) | 21 (6.9%) |
| UUID | 81 (26.6%) | 14 (4.6%) |
| GUID | 32 (10.5%) | 16 (5.2%) |
| Wi-Fi 情報 | 97 (31.8%) | 36 (11.8%) |
| GPS 情報 | 67 (22.0%) | 25 (8.2%) |
| メールアドレス | 51 (16.7%) | 17 (5.6%) |
| 電話番号 | 9 (3.0%) | 0 (0.0%) |
| 誕生日 | 11 (3.6%) | 2 (0.7%) |
| 性別 | 20 (6.6%) | 5 (1.6%) |

表 3 にあるように，ADID を送信しているアプリが 139 個（45.6%）と最も多いものの，Android ID を送信しているアプリも，87 個（28.5%）あった．また，IMEI，UUID，および，GUID を送信しているアプリも，それぞれ，21（6.9%），14（4.6%），および，16（5.2%）であった．Android ID，IMEI，UUID，および，GUID などは，利用者による初期化が容易でないので，端末の永続的なトラッキングに利用され得る．

### 6.2 識別子情報の送信について

さらに，端末の永続的なトラッキングに利用可能な識別子情報（Android ID，ADID 及び IMEI）を送信しているアプリについて，Google Play ストアのカテゴリ別に数え上げたものを表 4 に示す．

カテゴリごとに識別情報を送信しているアプリの数に差がある．特に ‘Art & Design’ のカテゴリにおいては，9 個のアプリが Android ID を送信しており，3 個のアプリが IMEI を送信していたりと，その数は他のカテゴリに比べて顕著である．

### 6.3 その他の情報の送信について

メールアドレス，誕生日，および，性別を送信しているアプリについて，Google Play ストアのカテゴリ別に数え上げたものを表 5 に示す．

**表 4** カテゴリ別アプリによる利用者情報の送信状況（1）

| カテゴリ | 利用者情報を送信するアプリ数 | | | | |
|---|---|---|---|---|---|
| | Android ID | ADID | IMEI | UUID | GUID |
| Art & Design | 9 | 6 | 0 | 1 | 3 |
| Games | 3 | 7 | 0 | 4 | 0 |
| Parenting | 3 | 9 | 1 | 0 | 1 |
| Health & Fitness | 8 | 3 | 0 | 2 | 0 |
| Photography | 5 | 6 | 0 | 0 | 2 |
| Libraries & Demo | 7 | 1 | 0 | 0 | 4 |
| Music & Audio | 5 | 4 | 1 | 1 | 1 |
| Tools | 6 | 6 | 0 | 0 | 0 |
| Beauty | 2 | 5 | 1 | 1 | 0 |
| Maps & Navigation | 3 | 4 | 1 | 0 | 1 |
| Personalization | 5 | 2 | 0 | 0 | 2 |
| Weather | 4 | 4 | 0 | 0 | 1 |
| Auto & Vehicles | 3 | 3 | 1 | 0 | 1 |
| Business | 2 | 4 | 1 | 0 | 1 |
| Comics | 1 | 5 | 0 | 1 | 1 |
| Education | 1 | 4 | 1 | 2 | 0 |
| Entertainment | 2 | 3 | 1 | 1 | 1 |
| Family | 2 | 3 | 1 | 1 | 1 |
| Shopping | 1 | 6 | 1 | 0 | 0 |
| Video Players & Editors | 3 | 4 | 1 | 0 | 0 |
| House & Home | 0 | 7 | 0 | 0 | 0 |
| News & Magazine | 0 | 4 | 3 | 0 | 0 |
| Productivity | 2 | 3 | 0 | 1 | 1 |
| Lifestyle | 1 | 4 | 0 | 1 | 0 |
| Book&Reference | 2 | 3 | 0 | 0 | 0 |
| Dating | 0 | 5 | 0 | 0 | 0 |
| Food&Drink | 2 | 3 | 0 | 0 | 0 |
| Medical | 1 | 4 | 0 | 0 | 0 |
| Social | 1 | 4 | 0 | 0 | 0 |
| Sports | 0 | 4 | 0 | 0 | 0 |
| Travel & Local | 1 | 3 | 0 | 0 | 0 |
| Communication | 1 | 2 | 0 | 0 | 0 |
| Finance | 0 | 3 | 0 | 0 | 0 |
| Events | 1 | 1 | 0 | 0 | 0 |

## 7. 考察

### 7.1 トラッキングの可能性

表 3 に示したとおり，Android ID や IMEI といった端末識別子が，サードパーティに送信されていた．Android ID は，OS に紐づく識別子であり，利用者によるファクトリーリセットを行わない限り初期化されない．さらに，IMEI は，利用者（契約）自体に紐づく識別子であり，利用者による初期化や変更は難しい．これらのことから，対象アプリの利用者はトラッキングをされる可能性があると言える．

Android Developpers のドキュメント [2] にあるベストプラクティスにおいて，Android ID や IMEI などの端末

表 5　カテゴリ別アプリによる利用者情報の送信状況（2）

| カテゴリ | 利用者情報を送信するアプリ数 | | |
|---|---|---|---|
| | メール | 誕生日 | 性別 |
| Art & Design | 3 | 0 | 0 |
| News & Magazine | 1 | 1 | 1 |
| Book & Reference | 2 | 0 | 0 |
| Communication | 1 | 0 | 1 |
| Entertainment | 1 | 0 | 1 |
| Events | 1 | 1 | 0 |
| Libraries & Demo | 2 | 0 | 0 |
| Business | 1 | 0 | 0 |
| Comics | 1 | 0 | 0 |
| Health & Fitness | 1 | 0 | 0 |
| Medical | 0 | 0 | 1 |
| Personalization | 1 | 0 | 0 |
| Shopping | 1 | 0 | 0 |
| Tools | 1 | 0 | 0 |
| Video Players & Editors | 0 | 0 | 1 |

識別子の使用を避けることを原則としている．多くの利用者を抱える人気アプリでも，この原則に従わないものがあった．

端末識別子とは別に，メールアドレスをサードパーティへ送信しているアプリも少なからず存在していることがわかった．メールアドレスは，利用者に紐付く識別子となる．すなわち，複数の端末を持ち，それらの端末において同じメールアドレスを使用している利用者をメールアドレスを利用することで，端末を跨いだトラッキング，いわゆるクロスデバイストラッキングが行うことが可能となる．

### 7.2　位置情報の送信

GPS や Wi-Fi に関する情報をサードパーティへ送信しているアプリが存在していることがわかった．GPS 情報はもちろんであるが，Wi-Fi の情報のみでも，モバイル端末の位置を特定できることが示されている [1]．

端末識別子やメールアドレスと組み合わせることで，利用者がいつ，どこで，どのアプリを契機に，どのような行動をとったか，といった一連の行動をサードパーティから知られる可能性が考えられる．

## 8.　まとめ

本論文では，JDI ベースの Android アプリの動的解析機構，AADA を作成した．AADA を用いて，Google Play ストアの人気アプリ 305 個に対して動的解析を行い，アプリによる利用者情報の取得状況およびサードパーティへの送信状況を調査した．その結果，以下のことがわかった．

(1) Android ID や IMEI などの利用者を一意に特定できる可能性のある識別子がサードパーティに送信されていることがわかった．

(2) GPS や Wi-Fi の情報といった，位置情報に紐づく情報が送信されていることがわかった．

人気アプリにおいて，様々な情報を送信していることから，アプリ開発者はもちろん，サードパーティのライブラリ側も，扱う情報の種類を明示する仕組みが用意されることが期待される．

### 参考文献


[1] Tracking And Messaging Based On Prior SSIDs Received https://patents.google.com/patent/US20140169358A1/en.

[2] 一意の識別子のベスト プラクティス — Android Developers https://developer.android.com/training/articles/user-data-ids.

[3] Grace, M. C., Zhou, W., Jiang, X. and Sadeghi, A.-R. Unsafe Exposure Analysis of Mobile In-app Advertisements, Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC '12, New York, NY, USA (2012), ACM.

[4] Han, S., Jung, J. and Wetherall, D. A Study of Third-Party Tracking by Mobile Apps in the Wild.

[5] Kurtz, A., Hugo, G. and Tobias, B. Fingerprinting Mobile Devices Using Personalized Configurations, Proceedings on Privacy Enhancing Technologies (2016).

[6] Oracle, Java(tm) Debug Interface https://docs.oracle.com/javase/jp/8/docs/jdk/api/jpda/jdi/index.html.

[7] Seneviratne, S., Kolamunna, H. and Seneviratne, A. A Measurement Study of Tracking in Paid Mobile Applications, Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec '15, New York, NY, USA (2015), ACM.

[8] Wu, W., Wu, J., Wang, Y., Ling, Z. and Yang, M. Efficient Fingerprinting-Based Android Device Identification With Zero-Permission Identifiers, *IEEE Access*, **4** (2016), 8073–8083.

[9] アウンコンサルティング株式会社世界 40 カ国、主要 OS・機種シェア状況　【2018 年 3 月】～インバウンド Web プロモーションにシェア状況データを活用する～(2018), https://www.globalmarketingchannel.com/press/survey20180323.

[10] ジャスミンスマホ利用者に、アプリ利用状況を調査！アプリ課金状況や、インストール個数が明らかに (2018), https://honote.macromill.com/report/20180329/.

[11] 細谷竜平, 角田裕太, 森達哉, 齋藤孝道, モバイルアプリケーションが取得しているプライバシー情報の調査, コンピュータセキュリティシンポジウム 2017 論文集, 第 2017 巻 (oct 2017).

[12] 程斌, 角田裕太, 細谷竜平, 森達哉, 齋藤孝道, 動的解析による Android アプリが取得しているプライバシー情報の調査, 第 80 回全国大会講演論文集, 第 2018 巻 (mar 2018).

[13] 福田泰平, 明田修平, 瀧本栄二, 齋藤彰一, 毛利公一, JDWP による動的解析を利用した Android アプリケーションの外部モジュール利用実態調査, コンピュータセキュリティシンポジウム 2017 論文集, 第 2017 巻 (oct 2017).