Computer Security Symposium 2017
23 - 25 October 2017

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

福田 泰平[1]　明田 修平[1]　瀧本 栄二[1]　齋藤 彰一[2]　毛利 公一[1]

概要：Android アプリケーション (App) には，外部モジュールを組み込んだものが多い．外部モジュールには，個人に関する情報を無断で外部へ送信するものが存在し，これによって情報漏えいにつながる可能性が指摘されている．この実態を明らかにするために，JDWP を利用した動的解析ツールを構築し，マーケットに存在する App を対象に外部モジュールによって実際に外部へ送信された情報を観測した．その結果，ハードウェア識別子を送信する外部モジュールの存在や App が送信する個人情報の傾向が外部モジュールの利用状況に影響することが明らかとなった．本論文では，これらモジュールによる利用者情報の取得・送信状況の実態について報告する．

キーワード：モバイルセキュリティ，Android，プライバシ

A Study of the Usage of External Module on Android Applications using Dynamic Analysis Environment with JDWP

Taihei Fukuda[1]　Shuhei Aketa[1]　Eiji Takimoto[1]　Shoichi Saito[2]　Koichi Mouri[1]

Abstract: A lot of Android applications(apps) contain external modules that developed by third parties. However, some external modules send information in the smartphone without notice, which make issues of leakage privacy information. To reveal this issue, we built dynamic analysis environment using JDWP and analyzed apps in the official Android marketplace. In result, we understood existences and impacts of external modules sending hardware identifiers. In this paper, we report that the usage of privacy information by external modules.

Keywords: Mobile Security, Android, Privacy

1. はじめに

Android のアプリケーション (以下，App) には，App 開発者以外が作成した外部モジュールを組み込んでいるものが存在する．外部モジュールには，広告を表示するための広告モジュール，リリース後の運営支援に利用される監視モジュール，ゲーム開発の支援を行うゲームエンジンなどがある．App 開発者は，外部モジュールを利用することで，広告収入や開発工程の削減などの恩恵を受けることができる．

一方，外部モジュールの中には，端末内の利用者情報を収集するもの (以下，情報収集モジュール) が存在する．情報収集モジュールが扱うことのできる利用者情報には，Web の閲覧履歴，通話履歴，位置情報などのプライバシ情報に加えて，加入者識別 ID(IMSI)，端末識別 ID(IMEI)，MAC アドレス，電話番号などの端末識別情報 (グローバル ID) があり，これらの情報が情報収集モジュールによって不正に外部へ漏洩していることが問題となっている [1]．特に，グローバル ID は，ハードウェアにあらかじめ設定されたハードウェア識別子があるなど端末の利用者によってその値を変更できないという性質から，個人の特定につ

[1] 立命館大学
Ritsumeikan University
[2] 名古屋工業大学
Nagoya Institute of Technology

ながる可能性が高い．利用者情報の漏洩に対して，Google は，UUID，Instance ID[2]，Ad ID[3] の利用を推奨し，その他のグローバル ID を利用しないように規則を設けている [4]．また，ハードウェア識別子の一つである MAC アドレスを取得できないように API の仕様変更 [5] を行うなど Google も対策を行っている．しかし，規則や仕様変更を加味した App 設計であるかどうかは，App 開発者や情報収集モジュール提供者に依存しており，情報漏洩への影響に関して実態は定かではない．

本論文は，以上の問題についてその実態を明らかにするため，以下の 2 点について述べる．

- デバッグインタフェースを利用した，モジュール単位の API トレース機構の構築
- API トレース機構を利用した情報収集モジュールによるグローバル ID の外部送信に関する実態調査

本論文では，情報収集モジュールの特定や利用されるグローバル ID を確認するため，デバッグ情報をやり取りするためのプロトコルである Java Debug Wire Protocol(JDWP) を利用した API トレース機構を作成した [6]．この API トレース機構は，スタックトレース情報を利用することでモジュールによる API 呼出しを観測することができる．実態調査では，GooglePlay[7] の新着無料 App を 1 ヶ月間収集し，合計 1761 検体に対して，API トレース手法を利用して情報収集モジュールによるグローバル ID の送信状況や，App 開発者との関連について調査を行った．

本論文の実態調査により，以下のような実態が明らかとなった．

- 調査対象の約 28 %の App で情報収集モジュールによってグローバル ID が送信され，約 11 %の App で UUID，Instance ID，Ad ID の以外のグローバル ID が送信されていた．
- App による各グローバル ID の送信状況は，App に組み込まれた情報収集モジュールの利用状況に大きく影響されていた．
- App の利用状況などを監視する監視モジュールは，Android ID や IMEI を送信する傾向にあった．
- 見た目をカスタマイズする着せ替え App やウィジェット App は，外部モジュールによって Android ID，IMEI，MAC アドレスを送信するものが多かった．

本論文では，2 章で外部モジュールによる利用者情報の送信を観測するために作成した解析環境について述べ，3 章で解析環境を利用して公式マーケットに存在する App を対象をとした実態調査の内容と結果について述べる．4 章で，調査結果の考察について述べ，5 章で関連研究について述べ，最後 6 章でまとめる．

2. 解析環境

本論文の調査では，外部モジュールによる利用者情報の流出を確認するため，API トレースとスタックトレース情報を利用して App に含まれる外部モジュールの挙動を観察する環境を構築した．本章では，Java のデバッグインタフェースである JDWP を利用した API トレース機構の概要と，API トレース結果を利用して外部モジュールによるグローバル ID の送信の有無を明らかにする解析ツールについて述べる．

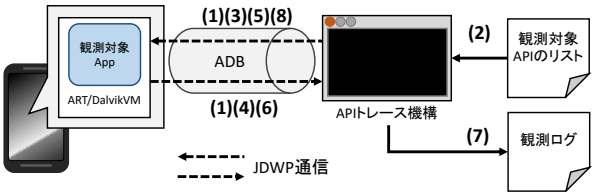

図 1　JDWP を利用した API トレース機構
Fig. 1　Mechanism of API tracing with JDWP

2.1 JDWP を利用した API トレース機構

本論文の調査では，外部モジュールの挙動を観測するため，App のデバッグに利用される JDWP を利用して API の呼出しを観測する．JDWP は，Android の Java VM である Android Runtime(以下，ART と記す) とデバッグ情報をやり取りするためのプロトコルであり，ソケット通信を介してデバッグ対象の App を観測することができる．

API 呼出しの観測は，API にブレークポイントを設定し，ブレークポイントの到達を検知することで行う．API 呼出しの観測時に取得する API ごとの詳細情報 (以下，API 情報と記す) は，ブレークポイント到達時にローカル変数，引数，返り値，インスタンスフィールドから取得する．API 情報でどのような値を取得するかについては 2.2 節にて後述する．API の呼出し元モジュールは，ブレークポイントに到達したスレッドが管理しているスタックフレーム情報から呼出し元モジュールのパッケージ名を取り出すことで特定する．

JDWP を利用した API トレース機構の具体的な動作を図 1 に示す．

(1) 観測対象の App を実行している ART と，API トレース機構を接続する．接続は，ADB の JDWP 通信をフォワーディングする機能を利用して行う．

(2) (1) による接続後，API トレース機構は，外部に保存された観測対象 API のリストを読み出す．観測対象 API のリストには，対象 API のクラス名とメソッド名，ブレークポイント到達時に取得する API 情報などが記述されている．

(3) API トレース機構は，(2) の情報をもとに，観測対象の API にブレークポイントを設定する．

(4) ART 上で実行している App のスレッドが (3) で指定したブレークポイントに到達すると，ART は，その

スレッドを停止させる．APIトレース機構は，到達したブレークポイントの位置 (クラス名や行数など) をARTから通知される．

(5) その後，APIトレース機構は，API情報を取得するために，API情報が格納されたローカル変数，返り値，インスタンスフィールドなどの値を送信するようARTに要求する．また，APIトレース機構は，(4) で停止したスレッドのスレッドオブジェクトによって管理されているスタックフレーム情報を送信するようにARTに要求する．

(6) APIトレース機構は，ARTが送信した (5) のローカル変数，返り値，インスタンスフィールドの値とスタックフレーム情報を受信する．

(7) APIトレース機構は，(4) と (6) で受信した情報をAPI呼出し観測ログとして出力する．

(8) APIトレース機構は，(4) で停止したスレッドを再開させるようARTに要求し，停止したスレッドが再開する．以後，(4) から (8) が繰り返されることによって，APIトレース機構は，対象となるAppで呼び出された観測対象APIの情報を観測し続ける．

2.2 観測対象 API

本論文の調査では，情報収集モジュールによる利用者情報の外部送信を観測するため，表1のAPIを観測対象とした．

A. 外部通信に関する API

情報収集モジュールによる外部との通信を観測するためには，通信内容と通信先の情報が必要である．そこで，ソケット通信や暗号化通信を行うAPIを観測対象とすることで，外部との通信を観測する．また，HTTPリクエストの送信に利用されるAPIや，AndroidのWebViewを使用してウェブページを表示するためのAPIも観測対象とする．Androidでは，暗黙的インテントによってURLを指定することで，ブラウザAppを起動することが出来る．そのため，Activity起動に使用されるAPIも観測対象とすることで，ブラウザAppに送られたURL情報を観測する．通信内容と通信先は，API呼出しの観測時に，APIの引数やローカル変数から取得する．

B. 利用者情報取得に関する API

情報収集モジュールによる利用者情報の外部への送信を確認するため，通信内容に利用者情報が含まれていることを確認する必要がある．そこで，利用者情報の取得に使用されるAPIを観測対象とすることで，モジュールによって実際に取得された利用者情報を観測する．取得された利用者情報は，APIの返り値から取得する．

C. スタックフレーム情報の紐づけに必要な API


```
{
  "time": "17:06:36.216",
  "reference": "java.lang.Thread",
  "method": "start",
  "signature": "()V",
  "pid": 0,
  "tag": "CALLED",
  "thread": {
    "id": "17900",
    "name": "pool−1−thread−1"
  },
  "api_infos": {
    "stack": [
      "java.lang.Thread.start",
      "java.util.concurrent.ThreadPoolExecutor.addWorker",
      "java.util.concurrent.ThreadPoolExecutor.execute",
      "bje.run",
      "java.util.concurrent.ThreadPoolExecutor.runWorker",
      "java.util.concurrent.ThreadPoolExecutor$Worker.run",
      "java.lang.Thread.run"
    ],
    "this.id": 17901
  }
},
```


図 2 APIトレースログの例
Fig. 2 Example of API tracing log.

スタックフレーム情報は，スレッド単位で存在する．そのため，1つのスタックフレーム情報だけでは，スレッドを超えた呼出し元モジュールを特定できない．そこで，スレッドの生成とスレッド間通信を行うAPIを観測対象とすることで，不足したスタックフレーム情報を取得する．

2.3 観測ログ

APIトレース機構は，観測対象APIの呼出しを観測したことを解析可能なログとして出力する．APIトレース機構が出力する観測ログの例を図2に示す．図2は，スレッドを生成するAPIであるThread.start() メソッドを観測したときのログである．観測ログには，APIのパッケージ名を含むクラス名 (reference) とメソッド名 (method) やAPIを呼び出したスレッドのTID(thread.id) などを記録する．さらに，API呼出し時のスタックフレーム情報 (api_info.stack) とAPI呼出し時のローカル変数やインスタンス変数の値を記録さする．API呼出し時のスタックフレーム情報からパッケージ名を確認することによって，どのモジュールがAPIの呼び出しに関与したか明らかにすることができる．

2.4 観測ログの解析

本論文の調査では，APIトレース機構から出力された観測ログを解析し，外部モジュールによる利用者情報の送信の有無を抽出する．観測ログの解析は，利用者情報の送信

表 1　観測対象 API
Table 1　Target APIs.

分類	API 名	取得する情報 (API 情報)
A	libcore.io.Posix.sendto	通信内容
A	com.android.org.conscrypt.OpenSSLSocketImpl$SSLOutputStream.write	通信内容
A	com.android.okhttp.internal.huc.HttpURLConnectionImpl.connect	HTTP リクエストの内容
A	org.apache.http.impl.client.DefaultRequestDirector. execute	HTTP リクエストの内容
A	android.webkit.WebView.loadUrl	URL
A	android.webkit.WebView.postUrl	URL，ポストされた情報
A	android.webkit.WebView.loadDataWithBaseURL	URL
A	android.app.Activity.startActivityForResult	ブラウザ App に送った URL
A	android.app.Activity.startActivityAsUse	ブラウザ App に送った URL
B	android.provider.Settings$Secure.getString	Android ID
B	android.telephony.TelephonyManager.getDeviceId	IMEI
B	android.telephony.TelephonyManager.getSubscriberId	IMSI
B	android.telephony.TelephonyManager.getSimSerialNumber	ICCID
B	android.telephony.TelephonyManager.getLine1Number	電話番号
B	android.net.wifi.WifiInfo.getMacAddress	MAC アドレス
B	com.google.android.gms.ads.identifier.AdvertisingIdClient$Info.getId	Ad ID
B	java.util.UUID.toString	UUID
B	com.google.android.gms.iid.InstanceID.getId	Instance ID
C	java.lang.Thread.start	生成されたスレッドの ID
C	android.os.MessageQueue.enqueueMessage	Message のオブジェクト ID

の確認と関与したモジュールの抽出の 2 つの段階を通して行う．

利用者情報の送信の確認

利用者情報の外部送信は，API トレース機構で観測した通信内容に利用者情報と一致する文字列があるか比較することで判定する．また，文字列比較を行う際には，利用者情報に対して MD5，SHA1，SHA256，SHA512 でハッシュ化した値も比較対象とする．ハッシュ化した値とバイナリ情報を含む通信内容との比較では，バイナリ情報を 16 進数文字列に変換したものを比較対象とする．通信内容は，表 1 の A で観測した通信内容や URL のクエリ内に含まれる情報を使用する．利用者情報は，表 1 の B で観測した実際に取得された利用者情報の値を使用する．

関与したモジュールの抽出

API 呼出しに関与したモジュールの抽出には，API 呼出し時に観測されたスタックフレーム情報に含まれるパッケージ名を取り出すことで行う．また，API を実行していたスレッドが生成された際のスタックフレーム情報 (表 1 の C を観測することで観測可能) もパッケージ名の抽出対象とする．

3. グローバル ID の外部送信の実態調査

本章では，2 章で述べた解析環境を利用して行った外部モジュールによるグローバル ID の外部送信の実態調査とその結果について述べる．

3.1 調査目的

本調査の目的は，App によるグローバル ID の利用傾向と App に組み込まれた外部モジュールとの関連を明らかにすることである．対象とするグローバル ID は，ハードウェアまたは Android システムに固有に割り当てられた端末の利用者によって変更することが困難な以下のグローバル ID を対象とした．

Android ID

Android システムに割り当てられる識別子である．Android の初期起動時にランダムに生成され，ファクトリーリセットを行うまで Android システムと一意に紐づく．

IMEI

携帯電話メーカによって端末に一意に割り当てられる識別子である．携帯端末を一意に識別することができる．携帯電話事業者が端末の接続拒否や機能停止を行う際に端末の識別に利用される．Android で取得するためには，READ_PHONE_STATE パーミッションが必要である．

IMSI

SIM カードに格納された識別子である．携帯端末の利用者を一意に識別することができる．携帯電話事業者が利用者を識別するために利用される．Android で取得するためには，READ_PHONE_STATE パーミッションが必要である．

ICCID

SIM カードに割り当てられた固有の識別子である．IMSI と同様に携帯端末の利用者を一意に識別することができる．Android で取得するためには，READ_PHONE_STATE パーミッションが必要である．

電話番号

SIM カードに格納された電話番号は，端末を一意に識別するために使用できる．Android で取得するためには，READ_PHONE_STATE パーミッションが必要である．

MAC アドレス

NIC に一意に割り当てられる物理アドレスである．NIC の識別により端末を一意に識別することが可能である．Android のバージョン 6.0 以降では，Android フレームワークを使用して得られる MAC アドレスは定数化されており，App から取得することができない．

また，Google が外部送信での利用を推奨している以下のグローバル ID も調査対象とした．

UUID

任意のソフトウェア上で生成できるグローバル ID である．時刻や乱数などを使用した様々な生成方法が存在する．App では，UUID を生成・保持することで，App がアンインストールされるまで App インスタンスの識別に利用することができる．

Instance ID

Android や iOS で動作する App で利用可能な App インスタンスを識別することができる識別子である．セキュリティトークンや認証機能などに利用するための API も用意されている．

Ad ID

GooglePlay 開発サービスから提供される広告用のグローバル ID である．ユーザによってリセットすることが可能である．

本調査では，これらのグローバル ID の利用傾向を明らかにするため，グローバル ID を App の内部で取得した検体数と外部へ送信した検体数を算出した．また，外部へ送信されたグローバル ID と外部モジュールとの関連を明らかにするため，外部モジュールごとに各グローバル ID 送信への関与が確認された検体数を算出した．さらに，これらの結果をもとに，グローバル ID の送信に対する外部モジュールの影響度や外部モジュールと App 提供者との関連を調査した．

3.2 データセット

本調査では，GooglePlay に存在する App によるグローバル ID の送信に関する実態を明らかにするため，公式マーケットに存在する複数の App を対象に調査を行った．本調査では，公式マーケットが設けた 32 のカテゴリの新着無料 App を隔週で 1 ヶ月間 (2017 年 7 月上旬〜8 月上旬) 取得し，合計 1761 検体を調査対象とした．

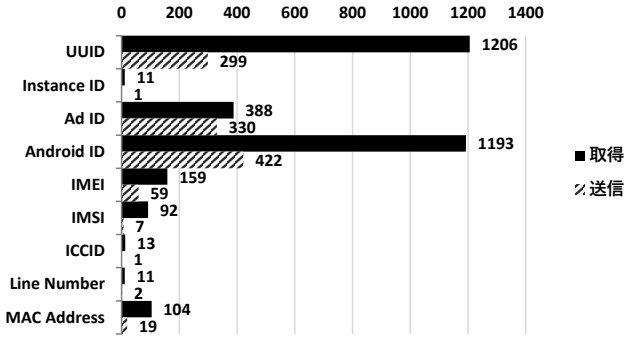

図 3 グローバル ID の取得と送信が確認された検体数
Fig. 3 Numbers of apps that fetch and send global identifiers.

本調査では，App 検体を Android 端末上でタイトル画面の出現を目安に数分間実行し，その間の API 呼出しを観測した．実行端末には，Android 6.0.1 を搭載した Nexus5 を使用した．App 検体の実行中に，画面に現れるランタイムパーミッションの要求ダイアログはすべて許可するようにタップ操作を行った．また，タイトル画面の前に出現する App 固有のログイン処理は，可能であればスキップするようタップ操作を行った．観測された API トレースログをもとに，グローバル ID の送信の有無と関与したモジュールを抽出し，その統計数を算出した．

3.3 調査結果

調査対象の 1761 検体における，各グローバル ID を API を用いて取得した検体数とその取得した情報を外部へ送信した検体数を図 3 に示す．多く取得されたグローバル ID は，UUID と Android ID の約 68 %であった．これらは，取得にパーミッションの要求が不要なため，端末を識別するのによく用いられるグローバル ID である．外部へ送信されたグローバル ID で最も多かったのは，Android ID であり 24 %と高い割合であった．Android ID は，Google の規則で外部への送信を控えるべきグローバル ID である．Android ID の代替として Google が提供した広告向けの Ad ID の送信検体数は 2 番目に多い 19 %であった．今回の調査では，Ad ID よりも Android ID の方が送信検体数が多く依然 Android ID が使用されている結果となった．

また，調査対象の 1761 検体に組み込まれていた外部モジュールにおける利用検体数と，外部モジュールによるグローバル ID の送信を確認した App 検体数を図 4 に示す．外部モジュールは，広告を表示する機能を持つ広告モジュール (Ad)，App 開発用のフレームワークを提供する開発エンジン (Engine)，App の利用傾向や不具合を監視する機能を持つ監視モジュール (Monitor)，画像処理，ログイン処理，データベース，通知機能などの開発を支援す

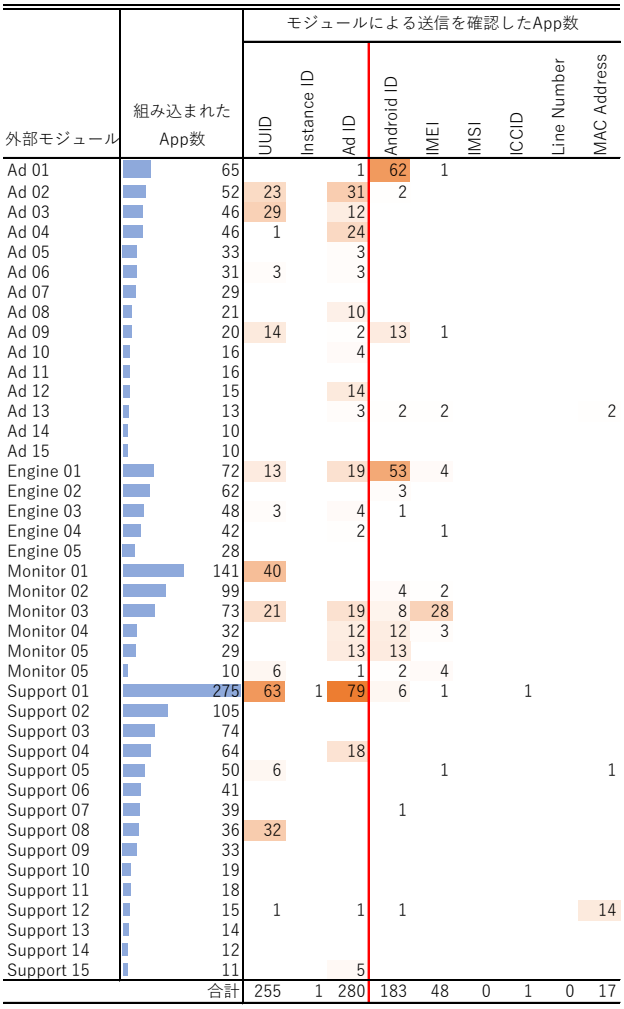

図 4 外部モジュールの利用検体数と各グローバル ID の送信検体数
Fig. 4 Usage of each external modules and numbers of found apps that send global identifiers.

る開発支援モジュール (Support) の 4 つに分類して示す．図 4 は，調査対象の 1761 検体の App の中で，10 検体以上の App に共通して組み込まれていた外部モジュールのみを示している．図 4 から重複した検体を除いた場合，外部モジュールによるグローバル ID の送信が確認された App は 501 検体，その内，UUID，Instance ID，Ad ID の以外のグローバル ID の送信が確認された App は 194 検体であった．

図 4 より，広告モジュールと監視モジュールは，グローバル ID の送信に関与する傾向にあった．特に，監視モジュールは，Android ID と IMEI を外部へ送信する傾向にあった．広告モジュールは，UUID や Ad ID を利用している傾向があるが，IMEI や MAC アドレスを利用しているものも存在していることがわかった．

開発支援モジュールは，モジュールの持つ機能によって利用するグローバル ID にばらつきがあるが，UUID や Ad ID を使用しているものが多いことがわかった．一方，Support 05，Support 07，Support 12 など Android ID，IMEI，MAC アドレスの送信に関与したモジュールも確認できた．

表 2 広告モジュールによって送信されるグローバル ID の変化
Table 2 Changes of global identifiers sent by advertisement modules.

広告	2016/1/18		2017/8/11	
	送信情報	個数	送信情報	個数
Ad A			Android ID	1
Ad B			UUID, Ad ID	1
Ad C	Ad ID	2		
Ad D	Ad ID	5		
Ad E	Ad ID	2	Ad ID	11
Ad F	Ad ID	3		
Ad G	Ad ID	9	Ad ID	2
Ad H	Ad ID	2	Ad ID, Android ID	3
Ad I	Ad ID	10	UUID, Ad ID	25
Ad J	Ad ID, IMEI	6	Ad ID, IMEI	2
	Android ID		Android ID	
	MAC Address		MAC Address	
Ad K	Android ID	8		
Ad L	Ad ID	1	Ad ID, MAC Address	1

*2015 年の調査では UUID を観測対象としていない

また，全てのカテゴリに対する新着無料 App 上位 50 個に組み込まれた広告モジュールについて，本調査と 2015 年に行った調査との比較を表 2 に示す．表 2 より，広告モジュールが送信するグローバル ID にあまり変化が見られず，依然 Android ID や MAC アドレスを送信する広告モジュールが存在していることが確認できた．

4. 考察

本論文の実態調査では，調査対象である 1761 検体のうち，約 28 %の App で外部モジュールによってグローバル ID が送信され，約 11 %の App で UUID，Instance ID，Ad ID の以外のグローバル ID が送信されていることがわかった．また，各グローバル ID を利用した App 数と各グローバル ID の送信への関与が確認された外部モジュールごとの App 数で図 3，図 4 のような傾向が明らかとなった．

図 3 では，利用が推奨されていない Android ID を利用する App が多いことが明らかとなった．Android ID の取得および送信が多かった背景として，Android ID はパーミッションを要求することなく取得することが可能なため，利用するための敷居が低いことが要因として挙げられる．さらに，Android における端末の識別子として選択肢が少ないことが挙げられる．iOS では，広告での使用を目的とした IDFA(Identification For Advertisers) とその他での使用を目的とした IDFV(Identifier For Vender) の 2 つのグローバル ID を提供している．IDFA はユーザの任意のタイミングでリセットが可能であり，IDFV は同一提供元の App をすべて削除すると変更されるグローバル ID である．IDFA は Ad ID に相当するグローバル ID であるが，IDFV のよ

表 3　App 提供元と Ad 01，Monitor 03，Support12 が組み込まれた App 数

Table 3　App's releasers and numbers of apps contains Ad 01, Monitor 03 and Support 12.

	Ad01	Monitor03	Support12	提供 App
提供元 01	1	1		着せ替え App
提供元 02	2	2		着せ替え App
提供元 03	3	3		着せ替え App
提供元 04	8	8		着せ替え App
提供元 05	2	1		着せ替え App
提供元 06		1		着せ替え App
提供元 07	1	1		着せ替え App
提供元 08		1		着せ替え App
提供元 09	1	1		着せ替え App
提供元 10	2			着せ替え App
提供元 11	8	4		着せ替え App
提供元 12		1		端末最適化 App
提供元 13		2		端末最適化 App
提供元 14	3	2		着せ替え App
提供元 15	31			Widget App，画面ロック App
提供元 16			14	着せ替え App

うなグローバル ID は Android で利用できない．Android においても，IDFV のような提供元単位のグローバル ID を設けることによって端末識別性の高いハードウェア識別子の利用を抑える効果が期待される．また，Android ID は，2017 年 8 月にリリースされた Android 8.0 において App 単位で値が変化する識別子に仕様が変更される [8] ため，今後の Android ID の利用傾向が注目される．

図 4 では，多くの外部モジュールで UUID と Ad ID が利用されているが，一方で監視モジュールの多くが Android ID と IMEI が利用している傾向が明らかとなった．図 4 において，各モジュールが送信するグローバル ID は同じ外部モジュールでも異なっている．このことから，App や組み込まれた外部モジュールのバージョンによって，外部モジュールの動作が異なることが推定される．

さらに，図 4 において，Android ID，IMEI，MAC アドレスの送信で特徴的な Ad 01，Monitor 03，Support 12 について詳細な解析を行った．その結果，App の提供元について表 3 のような特徴的な傾向が得られた．表 3 は，Ad 01，Monitor 03，Support 12 による Android ID，IMEI，MAC アドレスの送信が確認された App の提供元と提供されている App についてまとめたものである．表 3 より，Ad 01 は提供元 04，11，15 によって集中的に利用されていることがわかった．Monitor 03 も同様に提供元 04，11 によって利用されている傾向があった．特に，Support 12 を組み込んだ App はすべて提供元 16 によって提供されていることがわかった．また，表 3 に挙げた提供元の多くは，Android のホーム画面の見た目をカスタマイズするための着せ替え App を提供していた．さらに，Ad 01 を組み込んだ App を 31 個提供している提供元 15 も，ホーム画面上に気象情報を表示するウィジェット App を提供していた．以上のような結果は，見た目や機能の違いによって多くの着せ替え App やウィジェット App が提供されていることが要因と考えられる．このような着せ替え App やウィジェット App は，常に端末上で動作し続けるため，Android ID，IMEI，MAC アドレスによる個人特定のリスクが非常に高く危険である．

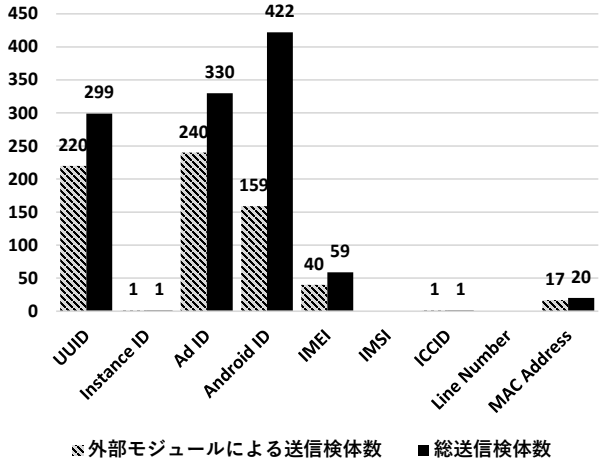

図 5　外部モジュールによる送信検体数と総送信検体数の比較

Fig. 5　Comparations of numbers of apps sending global identifier and related with external modules.

図 3 で示した各グローバル ID ごとの送信検体数と外部モジュールによるグローバル ID の送信検体数との比較を図 5 に示す．図 5 は，図 4 の各グローバル ID の送信合計数から，重複した検体を除いた数を用いている．図 5 より，Android ID 以外のグローバル ID は，半数以上が外部モジュールによって送信されていた．このことから，App が送信するグローバル ID の傾向は，組み込まれた外部モジュールに大きく影響していることがわかった．

本論文の実態調査では，App のタイトル画面が表示されるまでの期間が観測対象となっているため，外部モジュールが組み込まれているがグローバル ID の送信まで実行されなかった外部モジュールも存在していると考えられる．そのため，図 4 で示したグローバル ID の送信状況は最低限観測された数であり，その数以上の App がグローバル ID 送信のリスクを抱えている可能性がある．例えば，IMEI の送信が確認された開発支援モジュール Support 05 を組み込んだ 50 検体の App は，今回観測された 1 検体と同様に IMEI が送信されるリスクを抱えている．

5. 関連研究

App による情報漏洩を検出する TaintDroid[9] は，テイント解析技術を利用し，端末内の情報にタグを付与し，タグを追跡することによって外部へ情報が漏洩したことを

検出する．TaintDroid は，総務省が発行しているスマートフォンプライバシーアウトルック [10] で第三者が情報漏洩を検出する動的解析技術として取り上げられている．TaintDroid は，データの伝搬に着目しており，情報の送信に関与したモジュールを特定できない．本論文の調査では，API 呼出し時のスタックトレース情報を利用することで，グローバル ID の送信に関与したモジュールを正確に特定した．

Android のマーケットにおける，外部モジュールに関する研究に文献 [11], [12] があり，外部ライブラリに含まれる脆弱性の危険性について述べられている．文献 [11] では，パーミッションやモジュールの脆弱性が時間変化に伴いどのように変化するか調査されている．外部モジュールに含まれる脆弱性の変化を静的解析により明らかにすることで，外部モジュールの有無で脆弱性の変化に違いがあることを明らかにした．文献 [12] では，脆弱性が生じる要因を明らかにするため静的解析により外部モジュールに脆弱性が含まれているか調査されており，無料 App200 万検体中約 70 %，有料 App3 万検体中 50 %で外部モジュールによる脆弱性が確認された．

本論文の調査は，外部モジュールが送信するグローバル ID の現状・変化を調査するものであるため，スタックフレーム情報を利用した動的解析によって外部モジュールの動作を観測した．本論文の調査では，現在の外部モジュールによるプライバシ上の危険性について明らかにした．

6. おわりに

本論文では，外部モジュールに起因したグローバル ID の送信の実態を明らかにするため，Google Play で 1 ヶ月間に取得した 1761 検体の App に対して API トレースを利用した解析を行った．その結果以下の事実が明らかとなった．

- 調査対象である 1761 検体のうち，約 28 %の App で外部モジュールによってグローバル ID が送信され，約 11 %の App で UUID，Instance ID，Ad ID の以外のグローバル ID が送信されていた．
- App によって送信されるグローバル ID は，組み込まれた外部モジュールによる影響が大きかった．
- グローバル ID の中で最も外部へ送信された回数が多いものは，Android ID であった．
- IMEI，IMSI，MAC アドレスなどのハードウェア識別子を利用する外部モジュールが存在していた．
- App の利用率や不具合を監視する機能をもつ監視モジュールは，Android ID と IMEI を外部へ送信する傾向があった．
- 着せ替え App やウィジェット App は，外部モジュールによって Android ID，IMEI，MAC アドレスを送信するものが多かった．
- 広告モジュールが利用しているグローバル ID は 1 年前と比較して差がなかった．

以上のことから，現在においても外部モジュールによってハードウェア識別子が送信され，個人が特定されるリスクがあることが確認できた．また，特定の提供元によって Android ID，IMEI，MAC アドレスを送信する可能性のある外部モジュールが多用されていることが明らかになった．外部モジュールによる個人が特定されるリスクは，利用者からは判断できないため，App 開発者によるこれらのリスクに対する正確な理解が求められる．

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,“What is Instance ID? — Instance ID — Google Developers”, https://developers.google.com/instance-id/?hl=ja (2017)

[3] Google, “Android 広告 ID の使用 — 広告 — 収益化と広告 - Developer Policy Center”, https://play.google.com/intl/ja/about/monetization-ads/ads/ad-id/ (2017)

[4] Google,“Best Practices for Unique Identifiers — Android Developers”, https://developer.android.com/training/articles/user-data-ids.html (2017)

[5] Google,“Android 6.0 の変更点 — Android Developers”, https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html (2017)

[6] 福田 泰平，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 における JDWP を利用した 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)，No. 3D3-5，pp. 1-8 (2017)

[7] Google Play, https://play.google.com/store/apps, (2017)

[8] Google,“Android O での動作変更点 — Android Developers”, https://developer.android.com/preview/behavior-changes.html (2017)

[9] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B-G. Chun, L. P. Cox, J. Jung, P. McDaniel and A. N. Sheth, TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, ACM Transactions on Computer Systems (TOCS), Vol.32 No.2, pp.1-29 (2014)

[10] 総務省,“スマートフォン上のアプリケーションにおける利用者情報の取扱いに係る技術的検証等の諸問題に係る実証調査研究　スマートフォンプライバシーアウトルック II”, スマートフォン アプリケーション プライバシーポリシー普及・検証推進タスクフォース (2015)

[11] V. F. Taylor and I. Martinovic, To Update or Not toUpdate: Insights From a Two-Year Study of AndroidApp Evolution, Proceedings of the 2017 ACM on AsiaConference on Computer and Communications Security, pp. 45-57 (2017).

[12] T. Watanabe, M. Akiyama, F. Kanei, E. Shioji, Y. Takata, B. Sun, Y. Ishi, T. Shibahara, T. Yagi, and T. Mori, Understanding the origins of mobile appvulnerabilities: a large-scale measurement study of freeand paid apps,Proceedings of the 14th InternationalConference on Mining Software Repositories, pp. 14-24(2017)