実行コード上の機能間距離に基づくAndroidアプリの個人情報送信機能の推定

西田 雅太† 岩本 一樹† 星澤 裕二†

†株式会社セキュアブレイン 先端技術研究所
〒102-0083 東京都千代田区麹町 2-6-7 麹町 RK ビル 4F
†{masata_nishida, kazuki_iwamoto, yuji_hoshizawa}@securebrain.co.jp

あらまし Androidのパーミッションモデルでは、アプリが必要とする機能一覧を提示するのみであるため、ユーザは当該機能のアプリによる利用用途を知ることは出来ない。そのため、個人情報にアクセスするパーミッションを持つアプリにおいて、個人情報がユーザの意図しないかたちで外部に送信されてしまうといったリスクが存在する。本稿では、個人情報を取得する機能と情報送信する機能が、アプリのコールグラフ上でどの程度の隔たりが存在するかという機能間距離に着目し、アプリの個人情報送信機能との関係について調査する。また、調査結果に基づき機能間距離をパーミッション以外のアプリ機能の推定手法として提案する。

Induction of Sending Private Information Based on Function Distance in Android Application Bytecode

Masata Nishida† Kazuki Iwamoto† Yuji Hoshizawa†

†Advanced Research Laboratory, SecureBrain Corporation
Kojimachi RK Bldg., 6-7 Kojimachi 2-chome, Chiyoda-ku, Tokyo, Japan
†{masata_nishida, kazuki_iwamoto, yuji_hoshizawa}@securebrain.co.jp

Abstract Android's permission model only shows the list of features required by the application to execute. Based on this alone, users have no way of easily knowing whether their private information is being sent out without their consent. This paper will discuss the relationship between applications that send private information out and the distances of the features based on the applications' call graphs. This paper also will propose the use of feature distances to induce the feature set of the application being analyzed.

1 はじめに

Android では、アプリケーションによる機能やリソースに対するアクセスをパーミッションで制御している。ユーザは、アプリケーションが必要とするパーミッションの一覧をインストール時などに確認することが出来る。しかし、アプリケーションが実際にどのように機能やリソースを使用するかについての情報は提供されない。

そのため、パーミッションの使用宣言により許可された機能やリソースが、ユーザの意図しない形で利用されてしまうことが起こりうる。

Android マルウェアのひとつである Android.Uranico[1]では、アドレス帳に登録され

た連絡先から相性の良いものを占いで選ぶという機能を謳い、アドレス帳にアクセスするパーミッションを要求していた。一方で、アドレス帳の情報を外部に送信する機能が実装されていた。このように、アプリケーションが謳っている機能に合致したパーミッションを別の目的で悪用した実例がある。

この例からも分かるとおり、個人情報に関するパーミッションの利用用途が明確にならない点は、Android 端末を利用するユーザにとってリスクとなる。

本稿では Android アプリケーションのコールグラフから 2 つの機能の距離(以降、機能間距離)の算出方法を提案する。また、アドレス帳情報の取得機能と情報送信機能についての機能間距離を、個人情報送信機能の有無を推定するための指標として提案する。

2 関連研究

Android のパーミッションに関する関連研究として、アプリケーション実行時にパーミッションの使用許可を動的に制御する提案がある。

川端ら[2]は、パーミッションに該当する機能がアプリケーションで使用される際にユーザに対して実行確認を行うことを提案している。これにより、ユーザが意図しないタイミングでのパーミッションで許可した機能の使用を防ぐことを目的としている。[2]の手法では、機能の使用タイミングをユーザが制御することができるが、機能の用途をユーザが明示的に知ることが出来ない点は解決されていない。

一方、アプリケーションのソースコードの AST(抽象構文木)から、パーミッション利用用途を明らかにする研究も行われている。坂下ら[3]は、AST をもとに API から取得した個人情報が、外部に情報送信する API まで到達するかどうか、変数やメソッド間の情報の伝播を解析することで、パーミッションの用途の可視化を試みている。

本稿では、坂本らの研究と同様に静的解析結果を用いるが、アプリケーションの実行コードを抽象化したコールグラフを用いて機能の利用用途を推定する手法を提案する。

3 機能間距離

3.1 コールグラフ

Android アプリケーションの実行コードである dex ファイルの仕様は公開されており[4]、仕様に基づき dex ファイルを解析することができる。

dex 内には、定義されているメソッドごとに code_item というセクションが存在し、code_item セクションの中にメソッドの実行命令列(バイトコード)が記述される。バイトコードの命令は、全部で 218 種類定義されており[5]、このうち invoke-kind 及び invoke-kind/range がメソッド呼び出しを行う命令である。

バイトコードの中からこれらのメソッド呼び出し命令を抽出することで、メソッド間の呼び出し関係をコールグラフとして描くことが出来る。

3.2 コールグラフと機能間の関連

コールグラフ上で、特定の 2 つの機能が共通の呼び出し元となるメソッドを持つ場合、その共通の呼び出し元のメソッドが呼び出された際の一連の処理の中で、2 つの機能が両方実行される可能性がある。そのため、2 つの機能の間に何らかの関連がある可能性があるといえる。

本稿における機能とは、任意の API 呼び出しや定数・文字列の参照などを指す。

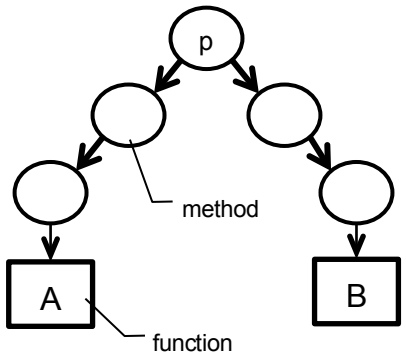

図 1 機能 A,B 間で関連のあるコールグラフ

図 1 のコールグラフではメソッド p が機能 A,B の共通の呼び出し元になる。メソッド p が呼ばれた際の一連の処理の中で、A,B 双方の機能が

実行される可能性がある。

一方、図 2 は機能 A,B 間に図 1 と同数のメソッドが存在しているが、A,B 間に共通の呼び出し元が存在しない。

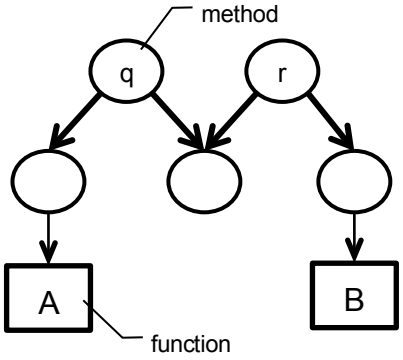

図 2 機能 A,B 間で関連がないコールグラフ

図 2 のコールグラフでは、機能 A の呼び出し元であるメソッド q が呼び出された場合に、機能 B が実行されることはない。同様に、機能 B の呼び出し元であるメソッド r が呼び出された場合にも、機能 A が実行されることはない。このように共通の呼び出し元を持たない 2 つの機能は、一連の処理の中で実行されることはない。

3.3 機能間距離

コールグラフ上で任意の 2 つの機能を呼び出している共通の呼び出し元を持つメソッドの距離を機能間距離として定義する。この際、グラフ上のメソッド間のエッジの重みを 1 とする。

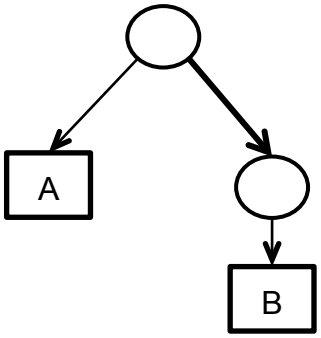

図 3 距離 1 の例

図 3 のように機能 A,B を呼び出しているメソッド間に 1 つのメソッド呼び出しが介在する場合、A,B の機能間距離は1となる。

また、図 4 のように対象の 2 つの機能 A,B が同じメソッドから呼び出されている場合には、A,B の機能間距離 0 となる。

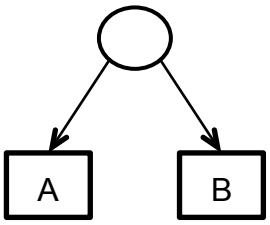

図 4 距離 0 の例

機能 A,B 間の機能間距離が小さい場合には、2 つの機能が近傍に位置しており、A,B が関連して実行される可能性が高くなると推測できる。

3.4 コールグラフの拡張

機能間距離の算出対象となる 2 つの機能が、違うスレッド上で非同期的に実行される場合、コールグラフ上に接点がなくなってしまい機能間距離が算出できない。そこで、本稿では下記のようにコールグラフの拡張を行い、非同期的に実行される機能の機能間距離も算出できるようにする。

Android SDK で用いられる非同期処理の開始メソッドと実行メソッドをコールグラフ上で接続する。その際のメソッド間の距離は 1 とする。

具体的には、AsyncTask を継承したクラスでは、execute メソッドの呼び出し箇所と doInBackground メソッドを接続し、Thread を継承したクラスでは、start メソッドの呼び出し箇所と run メソッドを接続する。

3.5 個人情報送信機能の推定

個人情報を取得する機能と、外部に情報送信する機能の機能間距離が算出可能なアプリケーションでは、これらの 2 つの機能が個人情報を外部に送信するために使用される可能性がある。また、当該機能間距離が小さい場合には、個人情報を送信する機能が存在する可能性が高くなることが推測できる。

以上より、個人情報を取得する機能と、情報送信機能の機能間距離から個人情報外部送信機能の有無を推定する手法を提案する。

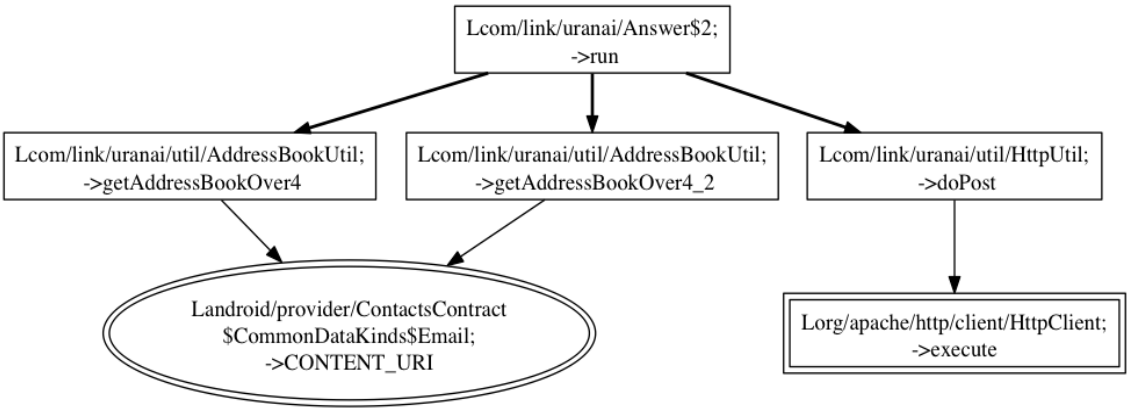

図 5 Android.Uranico の個人情報送信機能部分のコールグラフ

3.6 個人情報送信の機能間距離の実例

前述の Android.Uranico[1]のコールグラフを、個人情報を外部に送信するアプリケーションの例として示す。図 5 は Android.Uranico のコールグラフのうち、個人情報を外部へ送信している箇所を抽出したものである。

ContactsContract.CommonDataKinds.Email クラスの定数 CONTENT_URI を参照している箇所で、アドレス帳のデータベースからメールアドレスを取得している。また、HttpClient クラスの execute メソッドにより、HTTP 通信が行われる。

2 つの機能は、Thread を継承したクラスの run メソッドを呼び出した際の一連の処理の中で呼び出され、アドレス帳の情報が外部に送信される。図 5 のアドレス情報取得機能と外部送信機能の機能間距離は 2 となる。

4 検証

「3 機能間距離」の定義に基づき、実際の Android アプリケーションの個人情報を取得する機能と、外部に情報送信する機能に関する機能間距離を算出する。また、Android アプリケーションを手動で静的解析し、個人情報送信機能の有無を調査する。

4.1 検証用サンプル

著者らが Android アプリケーション配布サイトから独自に収集した、異なる証明書で署名されている 10,000 個の Android アプリケーションを検証用サンプルのベースとした。これらのアプリケーションは、2012 年 5 月から 2012 年 12 月までの期間で収集した。

検証用サンプルのベースのうち、アドレス情報の取得(READ_CONTACTS)とインターネット接続(INTERNET)のパーミッションの使用宣言しているアプリケーションの数を表 1 に示す。

表 1 検証用サンプルとパーミッション

Permissions	*Sample*
READ_CONTACT	776
INTERNET	8,750
READ_CONTACTS&INTERNET	731

(検証サンプルのベース 10,000 個中)

表 1 より、READ_CONTACTS と INTERNET の両方のパーミッションを持つアプリケーションは 10,000 個中に 731 個存在した。これらの 731 個のアプリケーションを検証用サンプルとする。

4.2 機能間距離の算出

4.2.1 検証対象機能

アドレス帳情報を取得する機能

アドレス帳情報の取得処理の際に特徴的なコードとして、アドレス帳機能に関連したクラスのクラス定数 CONTENT_URI の参照がある。

アドレス帳からデータを取得する際には、CONTENT_URI とクエリ文字列をパラメータとして ContentResolver オブジェクトから Cursor オブジェクトを取得する。Cursor オブジェクトから、アドレス情報を取得することが出来る。

アドレス帳にアクセスするための

CONTENT_URI が定義されているクラスの代表的なものとして以下のようなクラスがある。

- android.provider.Contacts
- android.provider.Contacts.Phones
- android.provider.ContactsContract.Contacts
- android.provider.ContactsContract.CommonDataKinds.Email

検証では、アドレス帳関連のクラスで定義されている定数 CONTENT_URI、36 個を検証対象とした。

情報を外部に送信する機能

HTTP または Socket 経由で外部に情報を送信する機能を検証対象の機能とし、以下の 7 つのメソッド呼び出しを検証対象とした。

- DefaultHttpClient#execute
- HttpClient#execute
- AndroidHttpClient#execute
- WebView#postUrl
- HttpURLConnection#getOutputStream
- URLConnection#getOutputStream
- Socket#getOutputStream

HttpClient インターフェースと HttpClient を実装したクラスの execute メソッドは、HTTP 通信の実行時に呼ばれるメソッドである。また、URLConnection や Socket クラスの getOutputStream メソッドは、HTTP の POST メソッドや Socket で直接データを送信する際などにデータの出力先を取得するメソッドである。

4.2.2 機能間距離の算出

前節までに示した条件のもとで、検証用サンプルの機能間距離の算出をしたところ、731 個中 146 個のサンプルのコールグラフで距離を算出することが出来た。

表 2 機能間接点の有無

	Sample	*Rate*
機能間接点あり	146	19.97%
機能間接点なし	585	80.03%
Total	731	100.0%

以降、機能間距離が算出できたものを機能間接点があるという。

機能間接点が存在したサンプルのうち、最小機能間距離ごとにサンプル数を集計したものを表 3 に示す。

表 3 最小機能間距離ごとのサンプル数

Distance	*Sample*	*Rate*
0	0	0.00%
1	5	3.43%
2	27	18.49%
3	20	13.70%
4	13	8.90%
5	24	16.44%
6	5	3.43%
7	17	11.64%
8	21	14.38%
9	5	3.43%
10	2	1.37%
11	2	1.37%
12	2	1.37%
13	1	0.68%
14	0	0.00%
15	0	0.00%
16	2	1.37%
Total	146	100.00%

算出できた機能間距離において、最小距離は 1、最大距離は 16 であった。

4.3 静的解析

機能間接点が存在した 146 個のサンプルと、機能間接点が存在しなかったサンプルの一部を手動で静的解析し、個人情報送信機能の有無を確認した。

具体的には、アドレス帳から取得した情報（メールアドレス、電話番号）が、HTTP や Socket を介して外部に送信されるかどうかについて解析した。解析には Android デコンパイラ JEB[6]を使用した。

4.3.1 機能間接点あり

機能間接点が存在したサンプルの静的解析結果を表 4 に示す。機能間接点が存在したサンプルでは、約半分のサンプルで個人情報送信機能が確認できた。機能間距離と静的解析結果の関係については、「4.4 機能間距離と静

的解析結果」で述べる。

表 4 機能間接点ありの静的解析結果

	Sample	*Rate*
Upload	75	51.37%
Not Upload	71	48.63%
Total	146	100.00%

4.3.2 機能間接点なし

機能間接点が存在しなかったサンプルのうち、難読化されていないものを中心に 40 個のサンプルの静的解析を行い、個人情報送信機能の有無を確認した。解析結果を表 5 に示す。

表 5 機能間接点なしの静的解析結果

	Sample	*Rate*
Upload	7	17.50%
Not Upload	33	82.50%
Total	40	100.00%

機能間接点が存在しなかったサンプルでは、約 20%弱のサンプルで個人情報送信機能が確認できた。

4.3.3 機能間接点と個人情報送信機能

機能間接点の有無と個人情報送信機能の関連を見るために、表 4 と表 5 を比較する。

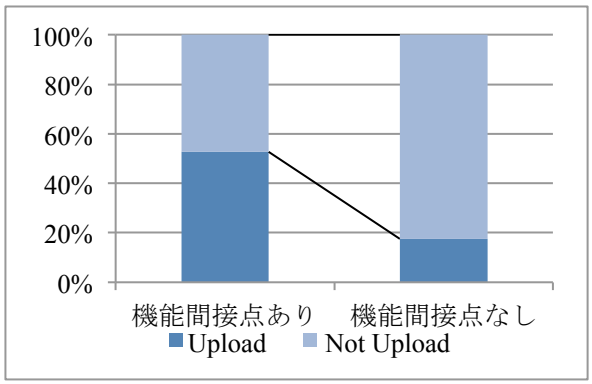

図 6 機能間接点の有無と情報送信機能

それぞれの個人情報送信機能の割合を比較した結果を図 6 に示す。

図 6 のとおり、機能間接点がある、すなわち機能間距離が算出できたサンプルのほうが、高い割合で個人情報送信機能が確認された。

4.4 機能間距離と静的解析結果

4.4.1 機能間距離ごとの個人情報送信機能

表 4 の結果を最小機能間距離ごとにまとめたものを図 7 に示す。

ただし、これらのサンプルにおいて、個人情報送信機能が確認されたのは最小距離のコールグラフとは限らず、他の機能間接点で個人情報の送信が確認されたものも含まれる。また、機能間接点では個人情報送信機能が確認されなかったものの、別の箇所で個人情報送信機能が確認されたサンプルも含まれている。

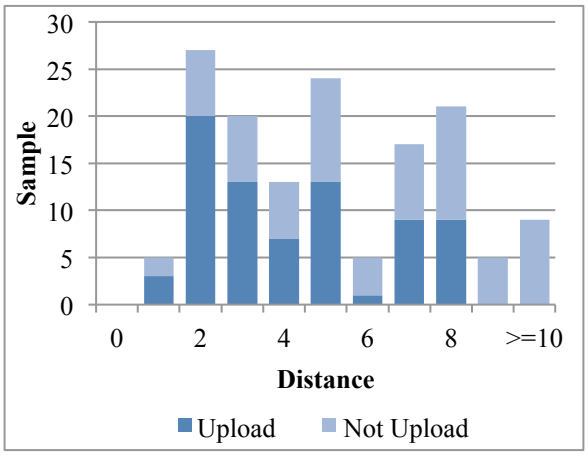

図 7 機能間距離ごとの個人情報送信機能

距離 2 のサンプルで最も多くの個人情報送信機能が確認できた。また、距離 9 以上では個人情報送信機能を有しているものは確認できなかった。

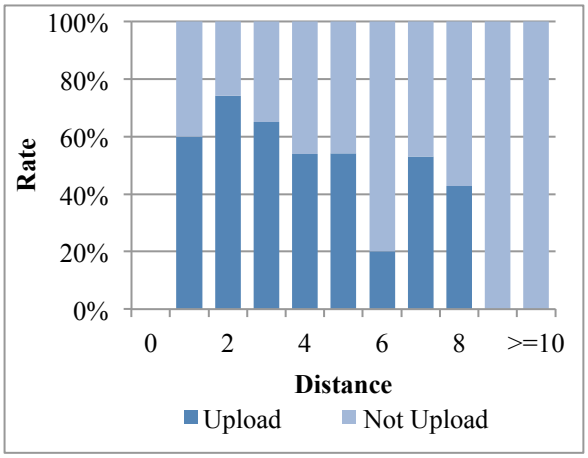

図 8 個人情報送信機能の距離別割合

図 7 を個人情報送信機能が確認できた割合に変換したグラフを図 8 に示す。個人情報送信機能の割合においても、距離 2 のサンプルが一番高い値を示した。

4.4.2 重複コールグラフの除外

図 7 の結果では、広告表示用ライブラリなど同一のライブラリによって個人情報送信機能を実現しているサンプルが複数存在した。そこで、複数使用されているライブラリの影響を除外するために、個人情報送信機能を有する同一のライブラリを使用しているサンプルをまとめて 1 つのサンプルとカウントすることで、重複したサンプルを除外した結果を図 9 に示す。

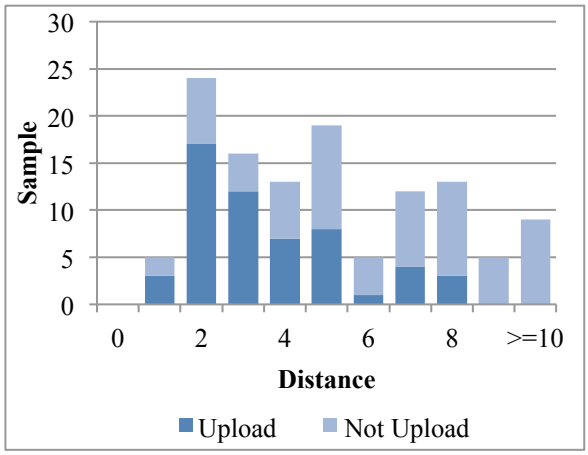

図 9 機能間距離ごとの個人情報送信機能 (重複サンプル除外)

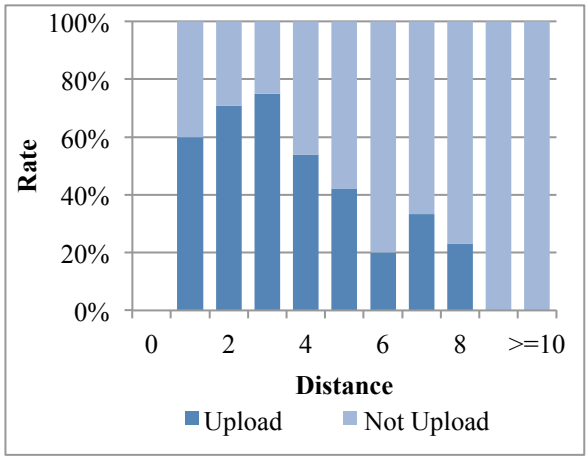

図 10 個人情報送信機能の距離別割合 (重複サンプル除外)

同様に、図 8 から重複したサンプルを除外した結果を図 10 に示す。

図 10 では、距離 3 を頂点として、機能間距離が増えるにつれて個人情報送信機能の割合が減少している様子が、図 8 よりもはっきりと見て取れる。

5 考察

5.1 検証結果

5.1.1 機能間距離の算出の有効性

図 6 に示したとおり、機能間距離が算出できたサンプルのほうが、個人情報送信機能が確認できた割合が高かった。これより、機能間距離の算出が、個人情報送信機能の推定のために有効であると考えられる。

しかし、機能間接点がない場合でも個人情報送信機能が 20%弱のサンプルで確認されており、提案手法ですべての個人情報送信機能を抽出できるわけではない。

機能間距離の算出方法の精度の向上については、「5.2.1 コールグラフの改善」で考察する。

5.1.2 機能間距離と情報送信機能の関係

図 8 に示したとおり、機能間距離が小さいサンプルの方が、個人情報送信機能を有している傾向があった。特に、図 10 のように同一ライブラリなど重複したコールグラフの影響を排除した場合には、その傾向が強く見られた。

機能間距離のみで個人情報送信機能の有無を判断することは難しいが、検証のような静的解析結果に基づき、統計情報を指標として、個人情報送信機能を推定することは可能である。例えば、機能間距離が 2 のアプリケーションは約 75%の割合で個人情報送信機能が存在するというように数値を示すことができる。

ただし、マルウェア Android.Obad[7]にみられるような、Java のリフレクション機能を使用した難読化が施されたアプリケーションでは、コールグラフを正確に描画することが出来ない。そのため、提案手法を適用することはできなくなる。

5.1.3 静的解析の問題

検証対象には、難読化されたものや、高度に

構造化されたもの、メソッド数が 30,000 を超えるような実行コードが非常に大きなサンプルも存在しており、その全てを静的解析出来ていない。検証では、個人情報の送信に関係しそうな部分を中心に静的解析を行ったため、サンプルの個人情報送信機能を見落としている可能性がある。また、静的解析の結果として個人情報送信機能が認められたとしても、そのコードがアプリケーション内で使用されない可能性もある。

以上のことより、検証における静的解析の精度が高いとは言い切れない部分がある。

5.2 課題・検討事項

5.2.1 コールグラフの改善

機能間接点がないサンプルにおいて、20%弱の個人情報の送信が確認されたが、これらのサンプルではクラス変数を介して個人情報のやりとりをしているものが少なくなかった。

このように、コールグラフ上では表現できない情報の伝播として、ファイル、データベースへの情報の保存、Intent など Android フレームワークを介するものなどが考えられる。また、インターフェースメソッドの呼び出しとインターフェースメソッドの実装部では、コールグラフが接続されないといった問題がある。

これらの要素についても「3.4 コールグラフの拡張」と同様にコールグラフの拡張を行うことで、機能間距離が算出できるものが増加することが期待できる。

5.2.2 機能間距離の算出方法の最適化

本稿では、コールグラフ上のメソッド間の重みを1として定義したが、private メソッドなど同じクラス内でのメソッド呼び出しでは距離を小さく設定するなど、機能間距離を算出する際のパラメータをチューニングすることで、機能間距離の算出の最適化ができると考えられる。

6 おわりに

本稿では、コールグラフから機能間距離を算出し、Android アプリケーションの個人情報送信機能を推定する手法を提案した。検証では、機能間距離が小さいサンプルにおいて高い割合で個人情報送信機能が認められた。

本稿では、アドレス帳の情報送信に着目して検証を行ったが、他の機能や情報についても同様の手法が適用可能である。また、Android マルウェアの表層解析として、マルウェアが持つ機能の推定などにも応用ができると考えられる。

参考文献

[1] Android OS を標的とした不審なアプリに関する注意喚起, http://www.ipa.go.jp/security/topics/alert20120523.html

[2] 川端 秀明, 磯原 隆将, 竹森 敬祐, 窪田 歩, 可児 潤也, 上松 晴信, 西垣 正勝, Android OS における機能や情報へのアクセス制御機構の提案, CSS2011

[3] 坂下 卓弥, 小形 真平, 海谷 治彦, 海尻 賢二, 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, Vol.2013-CSEC-62 No.21

[4] Dalvik Executable Format, http://source.android.com/devices/tech/dalvik/dex-format.html

[5] Bytecode for the Dalvik VM, http://source.android.com/devices/tech/dalvik/dalvik-bytecode.html

[6] JEB - The Interactive Android Decompiler, http://www.android-decompiler.com/

[7] The most sophisticated Android Trojan, http://www.securelist.com/en/blog/8106/The_most_sophisticated_Android_Trojan