Vol.2014-CSEC-65 No.11
Vol.2014-IOT-25 No.11
2014/5/22

Android アプリの内部構造の区画化による個人情報保護の提案

小林真也[†1]　鈴木富明[†2]　可児潤也[†2]　川端秀明[†3]　竹森敬祐[†3]　西垣正勝[†2]

不正な Android アプリによる個人情報漏洩の問題を深刻にしている理由は，アプリ開発におけるプログラミングの自由度が高く，プログラム内の個人情報のデータフローを完全に把握することが難しいためである．そこで本稿では，プログラムの内部構造を「ゾーン（区画）」ごとに分離することによって，プログラム内での個人情報の伝搬範囲を限定する仕組みを導入する．開発者が，プライバシポリシに準じた形でアプリ内のクラスを区画化した上でアプリ開発することによって，個人情報は適切なゾーン内に囲い込まれる．この結果，各ゾーン間のメッセージパッシングのみを静的検査するだけで，アプリがプライバシポリシに準じていることが確認できるため，アプリとプライバシポリシの整合性（アプリがプライバシポリシ通りの動作をするか）検査も容易となる．

A Proposal of Privacy Protection by Separation Package of Android Application

SHINYA KOBAYASHI[†1] TOMIAKI SUZUKI[†2] JUNYA KANI[†2]
HIDEAKI KAWABATA[†3] KEISUKE TAKEMORI[†3] MASAKATSU NISHIGAKI[†2]

Recently, sensitive information leakage by Android malicious applications becomes more and more serious. This is due to difficulty of data-flow analysis in Android applications. In this paper, we introduce a mechanism to limit the propagation range of sensitive information in an Android application codes by properly separating the internal structure of the codes into “zones”. As a result, it can be confirmed that the Android application is in accordance with the privacy policy just by static analysis of the message passing between each zone without sophisticated data-flow analysis. Therefore, consistency checking between Android application and privacy policy would also become easier.

1. はじめに

近年の不正アプリによる端末情報（プライバシ情報）の漏えいが数多く報告されていることを受け，最近ではモバイル端末アプリに対してもプライバシポリシを表示することが総務省により推奨されている[1]．これとともにアプリマーケットがアプリのプライバシポリシ作成を支援し，第三者機関としてアプリのプライバシポリシを検査するような動きも始まっている[2]．

しかし，アプリ開発者がアプリのプログラムとプライバシポリシを別々に作成する形になっている以上，開発者の誤解やミスによって不正確なプライバシポリシが記述されてしまう可能性が残る．プログラムの静的／動的解析だけではアプリ内における端末情報（プライバシ情報）の利用状況を完全に把握することは容易でないため，端末側またはアプリマーケット側で，アプリとプライバシポリシの整合性を検査するには限界がある．

そこで本稿では，プログラムの内部構造を「ゾーン」と呼ばれる区画で分離することによって，プログラム内での端末情報の伝搬範囲を限定する仕組みを導入する．各ゾーンにおいては，それぞれ個人情報の取得と外部送信における制約が設けられている．開発者は，プライバシポリシに準じた形でアプリ内の各クラスを各ゾーンに分類した上でアプリ開発することによって，個人情報は適切なゾーン内に囲い込まれる．この結果，各ゾーン間のメッセージパッシングのみを静的検査するだけで，アプリがプライバシポリシに準じていることが確認できるため，アプリとプライバシポリシの整合性（アプリがプライバシポリシ通りの動作をするか）検査も容易となる．

本稿では，現在スマートフォンの世界市場において高いシェアを持つ Android OS を提案方式の適用対象として採用する．

2. プライバシポリシ

2.1 課題

モバイル端末アプリにおける端末情報（プライバシ情報）の取り扱いの健全化に対する社会的な要求を受け，総務省によりスマートフォンプライバシーイニシアティブが取りまとめられた[1]．その中で，スマートフォン向けアプリに対してもプライバシポリシ（アプリがどの情報を何の目的で使用するか）を表示することが推奨されている．しかし，プライバシポリシの運用にあたっては，下記の課題がある．

（１）　アプリ開発者側の課題：

アプリ開発者は，アプリを作成する際に，そのプライバシポリシについても記述しなければならない．すなわち，アプリ開発にあたっての負担がその分増大することになる（課題 1a）．また，アプリ開発の作業とプライバシポリシ作成の作業が分離されているため，アプリ開発

†1 静岡大学情報学部
Faculty of Informatics, Shizuoka University
†2 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
†3(株)KDDI 研究所
KDDI R&D Laboratories

の際にプログラムにバグが混入してしまった場合や，プライバシポリシ作成の際に誤りが混入してしまった場合などに，アプリとプライバシポリシの間の整合性（一貫性）が崩れることなる（課題 1b）．更には，不正なアプリ開発者が，実際のアプリの動作とは異なるプライバシポリシを故意に作成する可能性もある（課題 1c）．

（2） ユーザ側の課題：

ユーザがアプリを自身の端末にインストールする際には必ずプライバシポリシが表示されるようになっていたとしても，ユーザがその内容を理解できなかったり，十分に確認せずにインストールを許可してしまったりすることがある[1]（課題 2）．

（3） マーケット側の課題：

プログラムの静的／動的解析によってアプリ内における端末情報（プライバシ情報）の利用状況を完全に把握することは容易でない．このため，アプリマーケット側で，アプリとプライバシポリシとの整合性（アプリが本当にプライバシポリシ通りの動作をするのかどうか）を確認することが難しい（課題 3）．同様の理由で，アプリとプライバシポリシの整合性をユーザの端末内で検査することも容易ではない．

2.2 既存対策

課題 1a に対しては，アプリ開発者のプライバシポリシ作成を支援する方法が提案されている[4]．文献[4]では，チェックシート形式の入力方法を採用することによって，開発者がチェックシートにチェックを入れるだけでスマートフォンプライバシーイニシアティブに沿ったプライバシポリシが作成される仕組みを実装している．また，アプリ開発者にプライバシポリシ作成に対するインセンティブを与えるというアプローチも，課題 1a の対策として有効だと考えられる．文献[5]では，実際に，プライバシポリシを正しく記述しているアプリの開発者に対してアプリマーケットが奨励金を支払うという試みを運用した結果，プライバシポリシの充足率の向上が確認されたことが報告されている．

課題 1b，課題 1c の解決には，すべてのプライバシ情報取得 API に対してプライバシポリシに準じた機能制約を与えることにより，アプリ開発とプライバシポリシ作成を一体化させるというアプローチが提案されている[6]．しかし，スマートフォンアプリの利用目的は非常に多岐に渡るため，用意すべき機能制約 API の種類が無数に及んでしまい現実的な対策となり得ていない．

課題 2 に対しては，文献[4]の中で，ユーザにプライバシポリシを分かりやすく表示する工夫についても示されている．また，関連研究として，アプリに関するワーニング情報やパーミッション情報をユーザに分かりやすく説明するための方法が種々提案されている[7][8]．これら関連研究の知見は，プライバシポリシの表示に対しても活用できるのではないかと考えられる．

課題 3 に対しては，文献[9]が，アプリコードの静的検査によって広告モジュールやプライバシ関連 API の使用を確認し，プライバシポリシに記述するべき内容を提示する機構を提案している．また，アプリ内の端末情報（プライバシ情報）に「色付け」をすることによって，端末情報のトレースを可能にする方法が提案されている[10]．この技術を活用して，アプリにおける端末情報の利用がプライバシポリシに準じているかどうかの検査を端末側で動的に実施することができる可能性がある．

2.3 問題点

前述より，現時点では既存対策において課題 1b，1c における問題の解決に研究の余地があると考える． 課題 1b，1c については，アプリ開発とプライバシポリシ作成を一体化するアプローチが有用であるが，文献[6]のようにプライバシ情報取得 API のそれぞれに対してすべて個別に対応する方式は現実的ではない．そこで本稿では，アプリの内部をプライバシポリシに準じて区画化し，プライバシ情報を適切な「ゾーン（区画）」に封じ込めることによって，各ゾーン内でのプログラム開発の自由度を保ちながら，プライバシポリシと一体化したアプリ開発を可能とする方式を提案する．具体的な方法については次章で説明する．

3. 提案方式

3.1 コンセプト

Android OS では，端末番号や位置情報などのプライバシ情報ごとに情報取得用の API が用意されている．アプリは，必要な情報に対応する API をその都度呼び出すことによって，OS を介して当該プライバシ情報を入手することができる．アプリは，取得したプライバシ情報を利用して，ユーザにサービスを提供する．ここで，「アプリが，これらのプライバシ情報を，プライバシポリシに記された通りの方法によって適正に利用しているか」が重要となる．プライバシ情報取得 API の呼び出しを監視することによって，アプリがいつどのプライバシ情報を入手したかについては確認することは可能である．しかし，アプリに取得されたプライバシ情報は，その後，アプリ内で必要に応じて任意の形に加工されながら利用されるため，アプリ内のプライバシ情報の伝搬を完璧かつ効率的にトレースすることは困難である．

そこで本稿では，アプリの内部構造をプライバシ情報ごとに区画化して分離する開発方法を検討する．アプリ開発者がプライバシポリシに準じた形でアプリ内のクラスを区画化し，個々のプライバシ情報をそれぞれの区画（ゾーン）内に囲い込むことによって，各ゾーンの入出力さえ監視してやれば，アプリの挙動がプライバシポリシと整合しているか否かのチェックが可能になる．プライバシ情報は，ゾーンを超えての伝搬につながらない限り，ゾーン内でいか

に加工されようとも不問である．このため，アプリ開発におけるプログラミングの自由度も（プライバシ情報がゾーンを超えないという制約内で）維持される．

3.2 区画化ルール

厳密には，プライバシ情報ごとにアプリの内部構造を区画化する必要がある．しかし，ユーザが確認したい内容は「自身のプライバシ情報の内，どの情報がどのサーバに送られているか」であるため，プライバシ情報の外部送信先ごとにアプリの区画化をすれば十分である．このため，

- ゾーン 1：ゾーン内の情報を外部サーバ 1 に送信することが許可されているクラス群
- ゾーン 2：ゾーン内の情報を外部サーバ 2 に送信することが許可されているクラス群
- ：

という形で，複数のゾーンが規定されることになる．また，同時に，

- ゾーン 0：ゾーン内の情報をいかなる外部サーバにも送信することが許可されていないクラス群
- ゾーン∞：ゾーン内の情報を任意の外部サーバに送信することが許可されているクラス群

を規定する．

各ゾーンの送信制限は，具体的には，外部サーバ m を通信先として指定した形での情報送信用 API の呼び出しを，ゾーン m 内のクラス群に限定することによって実現される．ただし，ゾーン 0 内のクラス群は，外部サーバとの通信のための情報送信用 API の呼び出しは一切許可されず，ゾーン∞内のクラス群は，任意の情報送信用 API の呼び出しが許可される．

各ゾーンの情報取得は，アプリのプライバシポリシに基づいて制御される．例えば，「IMEI 情報をサーバ n に送信する」ことがプライバシポリシに記述されているアプリにおいては，IMEI 情報取得 API を呼び出すことができるのはゾーン n 内のクラス群に限られる．ただし，ゾーン∞内のクラス群は，すべてのプライバシ情報取得 API の呼び出しが一切許可されず，ゾーン 0 内のクラス群は，任意のプライバシ情報取得 API の呼び出しが許可される．なお，プライバシ情報に該当しない情報は，すべてゾーン∞に割り当てられる．

ゾーン間のメッセージパッシングに対しては，ゾーン n 内の任意のクラスとゾーン m（$n \neq m$）内の任意のクラスの間でのデータ送受信，ゾーン n 内の任意のクラスを引数に指定した形でのゾーン n 内の任意のクラスからのゾーン m（$n \neq m$）内の任意のメソッド呼び出し，戻り値を返す形でのゾーン n 内の任意のクラスからのゾーン m（$n \neq m$）内の任意のメソッド呼び出しのすべてを禁止する．ただし，ゾーン n 内の任意のクラスからゾーン 0 内の任意のクラスへデータ送信，ゾーン∞内の任意のクラスからゾーン m 内の任意のクラスへデータ送信，ゾーン n 内の任意のクラスを引数に指定した形でのゾーン n 内の任意のクラスからのゾーン 0 内の任意のメソッド呼び出し，戻り値を返す形でのゾーン n 内の任意のクラスからのゾーン∞内の任意のメソッド呼び出しについては許可される．

アプリの内部構造の区画化における上記のルールを図 1 にまとめた．図 1 のルールが守られる限り，アプリ内での各プライバシ情報はそれぞれ適切なゾーンの中に閉じ込められることになるため，アプリとプライバシポリシの整合が保証される．換言すれば，提案方式を利用して開発されたアプリに関しては，図 1 のルールが守られていることが確認できれば，そのアプリはプライバシポリシと整合していることが保証される．図 1 のルールのチェックは，アプリのソースコードがある場合はコードの静的検査によって確認可能である．このため，（アプリの安全性検証を徹底させるなどの目的で）アプリのソースコードの提出を義務付けているアプリマーケットなどにおいては，アプリのコードの静的検査という簡便な方法によって，プライバシポリシに対するアプリの整合性を確認することができる．

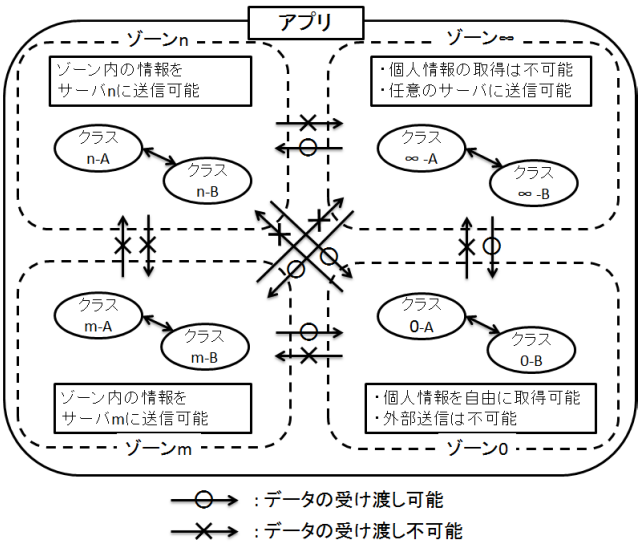

図 1 区画化ルール

4. Android アプリでの実装

本章では，提案方式を Android アプリの開発に適用する際の検討項目について示す．

4.1 課題

提案方式を Android で実現する際には，いくつかの課題が存在する．

① 情報の取得に関する問題
② 情報の送信先のセットに関する問題
③ ゾーンとクラスの紐付けに関する問題

①は，Android アプリのプライバシ情報取得 API 呼び出しにおける問題である．Android アプリでは Activity（アプリの画面を構成するコンポーネント）や Service（アプリをバックグラウンドで動作させるためのコンポーネント）と

呼ばれるコンポーネントが重要な役割を果たしており，アプリがプライバシ情報を取得する際には Activity または Service のインスタンスを必要とする．そのため，提案方式を Android アプリに適用する場合は，各ゾーン内に必ず Activity または Service を含まなければならないという制約が発生してしまう．本来であれば不要な Activity をアプリ内に用意することは無駄な画面遷移を生じ，本来であれば不要な Service をアプリ内に用意することは無駄なリソースの消費に繋がる．

②は，Android アプリの情報送信 API 呼び出しにおける問題である．Android アプリでは例えば，情報送信用の API として HttpClient.execute メソッドが用意されているが，通信先情報や送信内容等の各種パラメータ（HttpPost クラスのインスタンス）については，HttpClient.execute メソッドを呼び出す側で自由にセットすることができてしまうため，アプリのプログラムを静的解析するだけでは通信先を完全に特定しきれない場合がある．

③は，各クラスがどのゾーンに属しているかを開発者が明記しなければならないという問題である．

4.2 解決アプローチ

①に対しては，Android アプリのメイン Activity（またはメイン Service）のみを含む特別なゾーン（ゾーン M）を用意し，すべてのプライバシ情報取得 API をゾーン M 内に配置するという解決策を採る．その際，それぞれのプライバシ情報ごとに，Singleton パターン[11]を利用してプライバシ情報取得用のクラスを実装する．Singleton パターンをクラスの実装に適用することで，そのクラスのインスタンスがアプリ内でただ 1 つしか生成されないことを保証することが可能となる．すなわち，Singleton インスタンス内のプライバシ情報取得用 API を呼び出すことができるコンポーネントを，ゾーン M 内のメイン Activity（またはメイン Service）のみに限定することができる．

この Singleton パターンの性質を利用して，「アプリ起動時にプライバシ情報取得用 API を呼び出すことによって，すべてのプライバシ情報をそれぞれの単一インスタンスに格納し，以降，アプリ内の各ゾーンでそれらの情報が必要になった際には，当該インスタンス内の情報を参照する」よう開発に制限を加える．すなわち，Activity または Service を含まないゾーン内においては，プライバシ情報取得 API の呼び出しを行うべき部分で，当該 Singleton インスタンスに対する get メソッド呼び出しを実行する形となる．

Singleton インスタンスに対する get メソッド呼び出しは，アプリのプライバシポリシに基づいて制御される．例えば，「IMEI 情報をサーバ *n* に送信する」ことがプライバシポリシに記述されているアプリにおいては，IMEI 情報用 Singleton インスタンスの get メソッドを呼び出すことができるのはゾーン *n* 内のクラス群に限られる．ゾーン∞内のクラス群は，すべての Singleton インスタンスに対する get メソッドの呼び出しが一切許可されず，ゾーン 0 内のクラス群は，任意の Singleton インスタンスに対する get メソッドの呼び出しが許可される．

なお，Singleton インスタンスはプライバシ情報取得用 API と一対一対応した形で用意されることになるため，アプリのプライバシポリシに記されている情報については，Singleton インスタンスに関するコードを自動生成することが可能である．

②に対しても，①と同様に，送信先情報を格納した Singleton インスタンスをゾーン M 内に配置し，「アプリ内の各ゾーンで HttpClient.execute メソッド（情報送信用 API）が呼び出される際に，当該 Singleton インスタンスから送信先情報が既にセットされている HttpPost クラスのインスタンスを取得した上で，それを HttpClient.execute メソッドに直接セットする」という方法で対処する．

URL が http://hoge.com であるサーバ *n* に対し，HTTP ポストによって情報送信を行う場合の具体的な実装コード例を図 2a, 図 2b に示す. 図 2a がゾーン M（Singleton クラス）側のコード，図 2b がゾーン *n*（サーバ *n* に情報送信を行うクラス）側のコードである．

```
// ゾーン M(Singleton クラス)側
private final HttpPost post = new HttpPost(“http://hoge.com”);
public HttpPost getPost() {
    return post;
}
```

図 2a. 送信先情報の格納（HTTP ポストの場合）

```
// ゾーン n（サーバ n に情報送信を行うクラス）側
HttpClient httpClient = new DefaultHttpClient();
httpClient.execute(Singleton.getInstance().getPost());
```

図 2b. 送信先情報のセット（HTTP ポストの場合）

③に対しては，現時点では開発者がゾーンごとにパッケージを作成し，各ゾーン内のすべてのクラスを対応するパッケージ内で実装するという方法で対応する．さらに，「どのパッケージがどのゾーンのコードに対応しているか」に関する情報を，アプリのマニフェストファイル（AndroidManifest.xml）内に記述することとする．具体的には，ゾーンごとに「パッケージファイル名，取り扱われるプライバシ情報の種類，情報の送信先」を記載する形式となる．

開発者は，アプリのプライバシポリシを適切に記述した上で，実装する各クラスを適切なゾーンに分類しながらアプリの開発を進めることになる．

5. さまざまな攻撃への検討

提案方式を Android アプリに適用する上ではいくつかの抜け道が存在する．Android アプリ特有の機能や Java 言語特有の機能を用いることで，ゾーンを超えた自由な情報のやり取りが可能となってしまう．本章では，この問題に対

する対処として，アプリ開発に支障をきたさない程度のプログラミング上の制限を加えることを検討する．

5.1 リフレクション

Java にはリフレクションという機能が用意されており，これを利用することによって，通常であれば他のクラスからアクセスできない private や protected のフィールドやメソッドにアクセスすることが可能となる[12].

提案方式においても，このリフレクション機能を利用することでゾーン M 内に保持されているプライバシ情報を参照して，送信先情報を書き換えることが可能となってしまう．リフレクションは静的解析だけでは完全に検出することができないため，深刻なセキュリティホールとなる．リフレクションのサンプルコードを図 3 に示す．

```
import java.lang.reflect.Field;
Field field = instance.getClass().getDeclaredField(fieldname);
// フィールドの編集を有効にする
field.setAccessible(true);
// フィールドに値を設定する
field.set(instance, newValue);
```

図 3. リフレクションを使用したフィールドの変更例

この問題を解決するために，提案方式におけるアプリ開発においては，リフレクション機能のために用意されている Field クラスおよび Method クラスの setAccessible メソッドの使用を禁止するという制限を設ける．アプリマーケットにて実施されるコードの静的検査の結果，これらの使用が確認されたアプリに対しては，不正アプリと断定されアプリマーケットから削除される．

5.2 インテントによる値渡し

インテント（Intent）とは，Android アプリにおいて実装されているアプリ間およびコンポーネント間のメッセージパッシングの仕組みである[13]．送信側がインテントに任意の情報を格納して送信し，インテントを受け取った側でその情報を取り出して処理を行うことが可能となっている．コンポーネント間（アプリ内）でのインテントによるメッセージパッシングには，一般的にブロードキャストインテントが使用される．あるゾーン内のブロードキャストインテントに対応するブロードキャストレシーバを，他のゾーン内に作成することで，ゾーンを超えてインテントを受信することが可能となる．このため，プライバシ情報が格納されたブロードキャストインテントによって，プライバシ情報がゾーンの外に漏えいすることとなる．ブロードキャストインテント（送信側）とレシーバ（受信側）のサンプルコードを図 4a，図 4b にそれぞれ示す．

インテントは Android における重要な仕組みの 1 つであるため，この機能の使用そのものを禁止するわけにはいかない．そこで，この問題を解決するために，「インテント自体の送信は可能だが，インテントに任意のデータを格納した形でインテントを発信することは禁止する」という制限を加える．具体的には，インテントへのデータ格納メソッドである intent クラスの put*メソッドおよび set*メソッドの使用を禁止する．ただし，この制限によって，正規アプリの機能に悪影響が及ぶことがないか，詳細な確認が必要である．

```
// 個人情報を含むインテントの生成
String imei = Singleton.getInstance().getTEL ();
Intent intent = new Intent(“ACTION_GET_TEL”);
intent.putExtras(“tel”, tel);
// ブロードキャストインテントの送信
startBroadcast(intent);
```

図 4a. ブロードキャストインテント

```
// ACTION_GET_TEL アクションに対応する
// ブロードキャストレシーバ
public class BcReceiv extends BroadcastReveiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 個人情報の取得
        String tel = intent.getExtras().getString(“tel”);
    }
}
```

図 4b. ブロードキャストレシーバ

5.3 ファイル等を経由する値渡し

Android アプリには様々なデータの読み書きの機能が用意されている．SharedPreference，Content Provider，SQLite，ファイル I/O，SD カードの読み書きなどがこれにあたる．あるゾーンにおいて取得したプライバシ情報を一度ファイルやデータベースに保存した後で，別のゾーンまたはアプリから読み出すことによって，プライバシ情報をゾーン外に漏えいさせることが可能である．

もしプライバシポリシに関するアプリ開発者の手間を増やすことが許容されるならば，(プライバシ情報の送信先に加え）プライバシ情報の保存先についてもプライバシポリシに記述してもらうことでこの問題に対応できる．しかし，データの保存には様々な方法（クラスおよび各種メソッド）が存在するために，提案方式による実装や静的解析が複雑になることが懸念される．

5.4 その他

メモリ共有やネイティブコード連携，アプリ間連携を利用するデータの受け渡しなど，この他にも様々な攻撃とその対策について検討の必要がある．

6. 考察

6.1 開発者への制約

提案方式では，プライバシ情報の存在可能範囲に応じてプログラムの内部構造が適切に区画化されている．このため，それぞれのゾーン内においてはプライバシ情報をトレースする必要がなく，その観点では，アプリ開発の自由度が維持されている．しかし，ゾーン間のプログラミングに

おいては，アプリ開発者が制約を受けるため，それだけ開発者への負担が増すことになる．この問題については，必要に応じて動的解析や Android OS またはフレームワークに手を加えて情報を監視するなどの手法を併用していくことで緩和できるのではないかと考えている．具体的な方策については今後の検討を進めて行く．

6.2 パッケージレベル以外の区画化

本稿では，アプリ内のプログラムの内部構造をパッケージレベルで区画化することによってプライバシ情報の伝搬範囲を制限する方法を説明した．この他にもコンポーネントレベルやクラスレベルなど，アプリの内部構造を様々なレベルで区画化するアプローチが考えられる．更には，アプリごとに利用できるプライバシ情報とその送信先を制限した上で，アプリ連携の仕組みを活用することによって複数のアプリ群が 1 つのアプリであるかのような振る舞いをするような「メタアプリ」方式による区画化も可能であると考えられる．

Android OS であれば，アプリ間のメッセージパッシングの方法が限定されているため，メタアプリ方式には，ゾーン（アプリ）を超える情報のやり取りの監視が簡素になるというメリットがある．また，Android OS では，サンドボックスやパーミッションなどのアプリを保護するためのセキュリティ機構が充実している．メタアプリ方式であれば，これらの既存機構によって各ゾーン（アプリ）が守られるという利点もある．

6.3 プライバシポリシを意識したアプリ開発

現状のアプリ開発では，開発者がプライバシポリシを記述するタイミングは自由であり，アプリ開発を始める前からプライバシポリシを定義しておくことは必須ではなく，アプリ開発を終えた後にプライバシポリシを記述しても問題ない．しかし，プライバシポリシを考慮しないアプリ開発が，意図しない個人情報の漏えいを生じてさせてしまう一因となってはいないだろうか．提案方式では，アプリのプライバシポリシに応じた形でプログラムの区画化が規定されるため，必然的にプライバシポリシありきでアプリ開発が行われる形になる．今後益々アプリ内での個人情報の取扱いが厳格に定義され，アプリに対するプライバシポリシの適切な記述が求められていく中で，はじめからプライバシポリシを意識したアプリ開発というものが求められていくものであると考えている．

7. おわりに

本稿では，アプリ開発の自由度をなるべく下げることなく，プライバシポリシに準拠したアプリであるかどうかを簡単な静的解析によって保証する枠組みを提案した．Android アプリを対象に，提案手法による具体的なアプリ開発について検討した．

本稿はコンセプト提案の段階であり，現時点においては，まだ簡素なアプリを対象とした評価しか行っていない．今後早急に現実レベルのアプリを対象とした提案方式の評価・検証を行っていく必要がある．また，今後の課題としては，開発者の負担を更に軽減する方法の検討，プライバシポリシからゾーン M 内の Singleton インスタンスを自動生成する方法の具体化，静的検査ツールの実装・評価，動的検査導入の検討などが挙げられる．

開発者の負担軽減については，各クラスの適切なゾーンへの自動分類や，ゾーン間のメッセージパッシングに関するルール違反があった場合に開発者に警告を発するような開発環境の構築などの導入を検討している．更に将来的には，ネイティブライブラリで当該機能を提供するなどの方法によって，開発者がモジュールをアプリに組み込むだけで提案方式の導入が済むような仕組みが考えられないか検討する予定である．

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 ー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ー」の公表":
http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000087.html
2) 総務省,"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提言「スマートフォン安心安全強化戦略」（案）に対する意見募集":
http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000111.html
3) 磯原隆将 川端秀明 竹森敬祐 窪田歩:” XML を用いたモバイルアプリのプライバシーポリシーの運用”，2013 年暗号とセキュリティシンポジウム，2013.01
4) 竹森敬祐 磯原隆将 川端秀明 窪田歩 高野智秋 可児潤也 西垣正勝:”アプリ/コンテンツ向けプライバシーポリシーの第三者検証フレームワーク ”，情報処理学会研究報告,2013-CSEC-62,2013.07
5) 磯原隆将 川端秀明 竹森敬祐 窪田歩:” XML を用いたモバイルアプリのプライバシーポリシーの運用”，2013 年暗号とセキュリティシンポジウム, 2013.01
6) 鈴木富明 小林真也 可児潤也 川端秀明 磯原隆将 竹森敬祐 西垣正勝, “メタ API : プライバシポリシに準じた機能制約を備えた API”，コンピュータセキュリティシンポジウム 2013, 2013.10
7) Adrienne Porter Felt, et.al.: “How to Ask For Permission”, In Proc. USENIX Workshop on Hot Topics in Security, 2012
8) Sanae Rosen, et.al.: “AppProfiler: A Flexible Method of Exposing Privacy-Related Behavior in Android Applications to End Users”, Proceedings of the third ACM conference on data and application security and privacy, ACM(2013), pp. 221-232
9) 磯原隆将 川端秀明 竹森敬祐 窪田歩:” Android アプリの静的解析を用いた利用 API と外部モジュール検知によるプライバシーポリシー作成支援機構”，情報処理学会研究報告,
10) 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, Canada, 2010
11) ＠IT > Java Agile > Java TIPS > クラスのインスタンスを 1 つに保つ（Singleton パターン）:
http://www.atmarkit.co.jp/fjava/javatips/075java007.html (2014.04.11)
12) Javaプログラミングのダイナミックス: 第2回リフレクション入門: http://www.ibm.com/developerworks/jp/java/library/j-dyn0603/ (2014.04.11)
13) Android の重要な機能、インテント:
http://thinkit.co.jp/article/921/1 (2014.04.11)