Vol.2015-DPS-162 No.45
Vol.2015-CSEC-68 No.45
2015/3/6

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（その2）

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

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個人情報の送信先に応じて「ドメイン」と呼ばれる単位に区画化し，プログラム内での個人情報の伝搬範囲を限定した形でアプリ開発を行う仕組みを導入する．開発者が，アプリケーションプライバシポリシに準じた形でドメインを分離しながらアプリ開発を行うことによって，個人情報は適切なドメイン内に囲い込まれる．この結果，各ドメイ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 Package Separation of Android Application (part 2)

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

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 “domains”. 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 domain without sophisticated data-flow analysis. Therefore, consistency check 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プライバシ情報（端末情報）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「ドメイン」と呼ばれる区画で分離することによって，プログラム内でのプライバシ情報の伝搬範囲を限定する仕組みを導入する[「その1」のCSEC論文を参考文献に加えて下さい]．

各ドメインにおいては，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る．開発者が，プライバシポリシに準じた形でアプリ内の各コンポーネントを各ドメインに分類した上で，そのドメインの制約に従ったアプリ開発をすることによって，個人情報は適切なドメイン内に囲い込まれる．この結果，各ドメイ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議論を進める．

2. アプリケーションプライバシポリシ

2.1 課題

モバイル端末アプリにおけるプライバシ情報（端末情報）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，アプリケーション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 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
†2 (株)KDDI研究所
KDDI R&D Laboratories

（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[3]．文献[3]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[4]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[5]．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[3]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，アプリのインストール時に当該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[6][7]．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[8]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，アプリ内のプライバシ情報に「色付け（テイント）」をすることによって，プライバシ情報のトレースを可能にする方法が提案されている[9]．この技術を活用して，アプリにおけるプライバシ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 課題解決に向けてのアプローチ

前述より，現時点では既存対策において課題 1b，1c，2，3 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[5]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「ドメイン（区画）」に封じ込めることによって，各ドメイ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[13]．この結果，各ドメイ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）検査も容易となり，課題 2，3 の問題についても改善される．提案方式の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

一般的に，モバイル端末用の 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，

Vol.2015-DPS-162 No.45
Vol.2015-CSEC-68 No.45
2015/3/6

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処理（メソッド）を区画化し，個々のプライバシ情報をそれぞれの区画（ドメイン）内に囲い込むことによって，各ドメイ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，ドメインを超えての伝搬につながらない限り，ドメイ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（プライバシ情報がドメイ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，

- ドメイン 1：ドメイ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メソッド群
- ドメイン 2：ドメイ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メソッド群
- ：

という形で，複数のドメインが規定されることになる．また，同時に，

- ドメイン 0：ドメイ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メソッド群
- ドメイン∞：ドメイ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メソッド群

を規定する．

各ドメインの送信制限は，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，ドメイン m 内のメソッド群に限定することによって実現される．ただし，ドメイン 0 内のメソッド群は，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，ドメイン∞内のメソッド群は，任意の情報送信用 API の呼び出しが許可される．

各ドメインの情報取得は，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ドメイン n 内のメソッド群に限られる．ただし，ドメイン∞内のメソッド群は，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，ドメイン 0 内のメソッド群は，任意のプライバシ情報取得 API の呼び出しのみが許可される．

ドメイン間のメッセージパッシングに対しては，ドメイン n とドメイン m（$n \neq m$）間におけるデータの送受信，引数有りのメソッド呼び出し，戻り値有りのメソッド呼び出しのすべてを禁止する．ただし，ドメイン n からドメイン 0 へデータ送信，ドメイン∞からドメイン m へのデータ送信，ドメイン n からのドメイン 0 内の任意のメソッドの引数有りの呼び出し，ドメイン n からのドメイン∞内の任意のメソッドの戻り値有りの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ドメイ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．

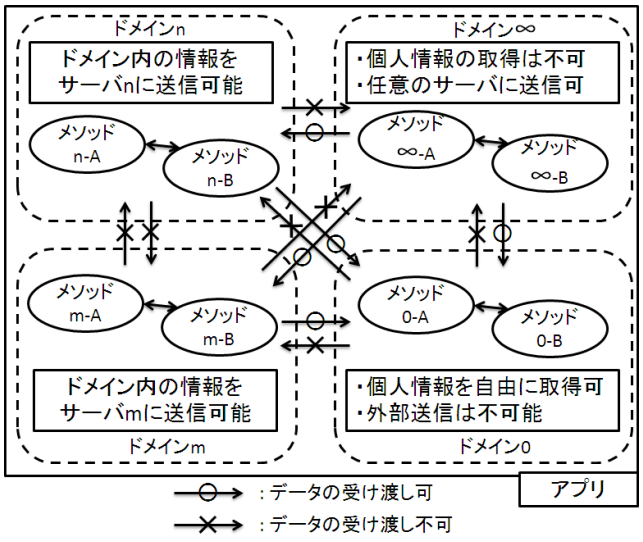

図 1. 区画化ルール

4. Android アプリでの実装

提案方式を Android アプリ開発に適用した際のアプリ開発方法について以下に示す．

Vol.2015-DPS-162 No.45
Vol.2015-CSEC-68 No.45
2015/3/6

4.1 開発フロー

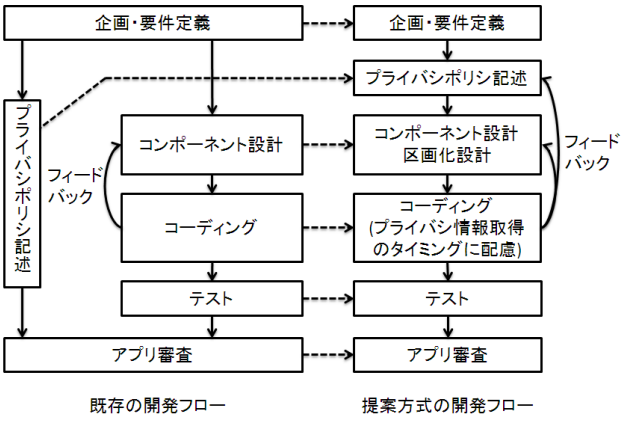

図 2. 提案方式におけるアプリ開発フロー

アプリ開発者がアプリをマーケットに登録するまでの一般的な開発フローは,「企画・要件定義→コンポーネント設計→コーディング→テスト→マーケット審査」という流れとなる．アプリケーションプライバシポリシの記述は，アプリ開発と並行して行われている．

提案方式においては，アプリ開発者はアプリの企画や要件定義を終えた後の設計工程で，アプリケーションプライバシポリシの記述を行う．これには，プライバシポリシを先に具体化することによって，アプリ開発者にプライバシポリシを十分に意識した設計および実装を促す意図がある．次に，アプリ開発者はコンポーネント設計の工程で，アプリの区画化の方針についても同時に設計を実施する（詳細は 4.2 節にて説明する）．その後，実装の工程では，アプリ開発者は，プライバシ情報取得 API の呼び出しのタイミングについても留意して，コーディングを進める（詳細は 4.3 節にて説明する）．実装の途中で仕様変更が生じた場合，その変更が上流工程にフィードバックされる．

図 2 にアプリの開発フローを図示した．

4.2 区画化の設計

Android アプリ開発においては，通常，Activity（アプリの画面を構成するコンポーネント）および 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）を単位として,プログラムの構造が設計されることになる．ここで，フォアグランドプロセスが「画面」を構成するのに対し，バックグラウンドプロセスは「隠し画面」と捉えることが可能である．この観点に立つと，Android アプリのコンポーネント設計は，アプリの画面遷移を決定することとほぼ等価となると考えられる．そこで，区画化のドメインもコンポーネント（画面）単位で設計する方針を採用することとする．

例として,「近隣レストランの情報を取得するために，端末の位置情報をデータベースサーバに送信する」というプライバシポリシを持つグルメアプリを考えよう．アプリ開発者がコンポーネント設計を行った結果が，図 3 の画面遷移であったとする．「メニュー画面」内の「近隣レストラン検索アイコン」をタップすると「近隣レストラン情報画面」に遷移し,「端末の位置情報のデータベースサーバへの送信→データベースサーバからの近隣レストラン情報の受信→受信結果の画面への表示」のタスクが実行される．

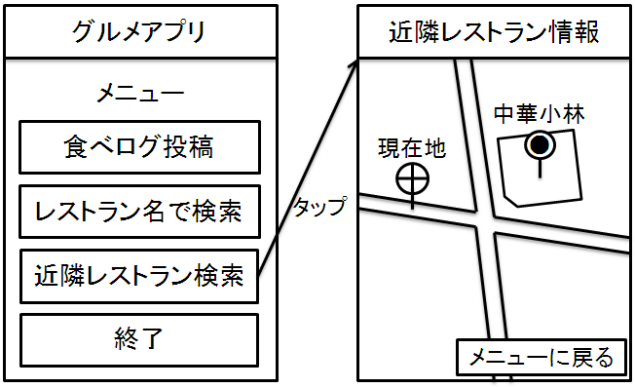

図 3. グルメアプリの例

ここで，プライバシ情報の送信に関係するタスクは,「近隣レストラン情報画面」内で実行される「端末の位置情報のデータベースサーバへの送信」である．そこで，提案方式においては,「近隣レストラン情報画面」に対応する Activity コンポーネントを「データベースサーバに送信する情報を扱うドメイン」として定義する．これにより,「データベースサーバに送信する情報」に該当する「端末の位置情報」を取得するための API 呼び出しは，この Activity（近隣レストラン情報画面）内でのみ許可されることになる．

Android アプリにおいては，多くの場合で，プライバシ情報取得 API の呼び出しの際にコンポーネントクラスのインスタンスを引数として指定する必要がある．コンポーネント単位で区画化を行うという設計方針は，この観点からも都合がよい．

一方で，提案方式においては，アプリ内の区画化に関する情報を何らかの形でアプリ自身に所持させる必要がある．Android アプリにおいては，Activity および Service コンポーネントを使用する場合には，必ずその情報をアプリのマニフェストファイル（AndroidManifest.xml）内に記述しなければならないという仕様となっている．コンポーネント単位で区画化を行うという設計方針であれば，マニフェストファイルにおける各コンポーネントの記述の中に，それぞれの区画の定義を追記するという方法を採ることができる．

マニフェストファイル内への区画（ドメイン）情報の記述には，”meta-deta”という要素を利用する．

記述例を図 4 に示す．図 4 の例では，MenuActivity というコンポーネントがドメイン 0 に属しており，NeighborSearchActivity というコンポーネントがドメイン”gourmet.com”に属していることが記述されている．また,

ドメイン”gourmet.com”から外部に送信可能な個人情報は位置情報（location）のみに限られており，送信先は”database.gourmet.com” のみに限られていることが記述されている．

```
<activity android:name=”com.hoge.MenuActivity”>
  <meta-data
    android:name=”domain”
    android:value=”0” />
</activity>
<activity android:name=”com.hoge.NeighborSearchActivity”>
  <meta-data
    android:name=”domain”
    android:value=”gourmet.com” />
  <meta-data
    android:name=”send.info”
    android:value=”location” />
  <meta-data
    android:name=”send.url”
    android:value=”database.gourmet.com” />
</activity>
…
```

図 4. マニフェストファイルにおけるドメインの定義

4.3 コーディングにおける留意

コーディングの工程では，それぞれの Activity および Service コンポーネントの「中身」のコードが作成されるが，各コンポーネントの中身のコード（java ファイル）の中で使用される全てのメソッドのドメイン情報は，原則としては，当該コンポーネントのドメイン情報がそのまま適用される．

ただし，プライバシ情報の外部送信は，それぞれのコンポーネントの中で「API 呼び出しを用いての当該プライバシ情報の取得→当該プライバシ情報の加工→当該プライバシ情報の外部送信」という流れで処理される．プライバシ情報取得 API が呼び出されるまではコンポーネント内にプライバシ情報は存在しないため，ドメイン n のコンポーネント内のメソッドであっても，当該 API が呼び出される前に処理を終えてしまうメソッドに対してはドメイン∞（プライバシ情報を含まないため，ドメイン内の情報を任意の外部サーバに送信して問題ない）と同等の扱いが可能である．

「プライバシ情報については，可能な限りこれを抱えこまない」という設計方針は，プライバシ保護の観点からも理に適っている．そこで提案方式では，各コンポーネントにおいて「プライバシ情報に関しては，外部送信メソッドに可能な限り近いポイントで情報取得 API を呼び出す」という点に留意してコーディングを行うよう，アプリ開発者に要請する．

4.4 整合性検証

3.2 節で述べたように，提案方式においては，図 1 のルールが守られていることが確認できれば，そのアプリがプライバシポリシと整合していることが保証される．提案方式を適用して開発された Android アプリのソースコードを受け取ったマーケットが，プライバシポリシとの整合性検証を行う際の具体的な手順を図 5 に示す．

Step1) マニフェストファイルからドメイン情報抽出
Step2) 全 Java ファイルを走査して以下をチェック
Step3-1) ドメイン 0 であれば
　1. 外部送信有 → 不正
　2. 他ドメインへのデータの送信有 → 不正
Step3-2) ドメイン∞であれば
　1. プライバシ情報の取得有 → 不正
　2. 他ドメインからのデータの受信有 → 不正
Step3-3) ドメイン n であれば
　1. ドメイン情報に指定されている送信先以外への外部送信有 → 不正
　2. ドメイン情報に指定されているプライバシ情報以外の取得有 → 不正
　3. ドメイン m との間でのデータの送受信有 → 不正
Step4) Step3 で不正がなければ適正アプリとして認定

図 5. 整合性検証の手順

4.5 制約

Android アプリの構成要素としては，Activity および Service のコンポーネントクラス以外にも様々なクラスおよびメソッドが存在する．現時点の提案方式においては，プライバシ情報の外部送信は Activity または Service コンポーネントのみが実行可能であるという制約をおき，Activity および Service のコンポーネントクラス以外のメソッドには自動的にドメイン 0 を割り当てるという前提をおく．

ブラウザの機能（WebView コンポーネント）を有するアプリケーションは，ユーザが閲覧先の URL を入力することによって，様々なサーバにアクセスする．したがって，提案方式の分類では，ブラウザ（Web View を含むコンポーネント）はドメイン∞に区画化される．しかし，その一方で，ブラウザアプリケーションは端末の位置情報などのプライバシ情報を利用することも可能である．すなわち，ドメイン∞のコンポーネントであるにも関わらず，プライバシ情報取得 API の呼び出しについてはこれを許可する必要があり，提案方式の区画化ルールに反する．このため，現時点の提案方式においては，ブラウザに代表される「汎用アクセス型」のアプリケーションに対しては提案方式の適用対象から除外する．

マーケットが実施するアプリとプライバシポリシの整合性検証に関しては，現時点では，開発者がアプリのソースコードをマーケットに提供することを前提としている．

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，ドメイ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，通常であれば外部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[10].

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，ドメイン n からドメイン m（$n \neq m$）内に封じ込まれているプライバシ情報を参照す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 6 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 6. 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[11]．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．あるドメイ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，他のドメイン内に作成することで，ドメイ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，プライバシ情報がドメイ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 7a，図 7b 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 7a. 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 7b. 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．あるドメイ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，別のドメインまたはアプリから読み出すことによって，プライバシ情報をドメイ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との連携，複数のアプリによる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. 既存アプリに対する区画化の適用と評価

6.1 My Tracks に対する区画化の適用

提案方式の実現可能性を確かめるために，既存アプリに対して提案方式の適用を試みる．今回は，Google 社がオープンソースとして公開している「My Tracks」というアプリケーション[12]を対象とした．My Tracks はアウトドアで活動した際のコースや速度，距離などのデータを記録し，Google ドライブで記録の同期や共有を行うことができる

サービスを提供している．

6.2 MyTracks が取り扱うプライバシ情報

My Tracks が取り扱うプライバシ情報は「位置情報」と「アドレス帳情報」の 2 種類である．位置情報は，アウトドアでの活動を記録するために利用される．アドレス帳情報は，記録した情報をメールで送信することによって他人と共有する際に，ユーザがメールアドレスを入力する際の入力支援としてオートコンプリート機能を提供するために用いられている．よって，MyTracks のプライバシポリシに記述される内容は Google ドライブに送信される位置情報についてのみであり，オートコンプリートに利用されるだけで外部へは送信されないアドレス帳情報についてはプライバシポリシに記述する必要はない．

図 8 に My Tracks のトラック共有設定時の画面を示す．左が Google ドライブへのエクスポートダイアログであり，右が Google ドライブでデータを共有する相手のメールアドレスを入力するダイアログである．提案方式では，プライバシポリシに従い，位置情報を扱うエクスポートダイアログ（図 8 左）がドメイン”Google.com”として，アドレス帳情報を扱うアドレス入力ダイアログ（図 8 右）がドメイン 0 として定義されることになる．

図 8 . My Tracks のエクスポート/共有画面

6.3 エクスポートダイアログのコーディング

MyTracks において，位置情報（トラック）を記録する機能は，TrackRecordingService というサービスコンポーネントにおいて実装されている．トラック情報は，MyTracksProvider と呼ばれるコンテンツプロバイダコンポーネントにおいてデータベースに保存され，SendDriveActivity というアクティビティにおいてデータベースから Google ドライブに送信される．したがって，これらの Activity と Service のコンポーネントについては，全てドメイン”Google.com”として定義し，その旨がマニフェストファイルにも記述される．

アプリ開発者は，”Google.com”以外のドメインからドメイン “Google.com”への値の受け渡しが禁止されていることに注意してコンポーネントの「中身」をコーディングしていく．例えば，Google ドライブへのトラック情報の送信は SendDriveAsyncTask と呼ばれる非同期タスク内で行われているが，SendDriveAsyncTask がドメイン”Google.com”の外に出ないように，SendDriveAsyncTask の実装は SendDriveActivity 内で行う必要がある．

6.4 アドレス入力ダイアログのコーディング

MyTracks において，メールアドレスのオートコンプリート機能は，ShareTrackDialogFragment というクラスにおいて実装されている．これは，AbstractSendToGoogleActivity というアクティビティの抽象クラスから呼び出されており，SearchListActivity，TrackDetailActivity，TrackListActivity という 3 つのアクティビティがこの抽象クラスを継承している．したがって，これらのアクティビティについては，全てドメイン 0 として定義し，その旨がマニフェストファイルにも記述される．ドメイン 0 のコンポーネントは外部送信メソッドを持たないので，プライバシ情報の取り扱いに注意を払う必要はなく，自由度の高いコーディングが許される．

6.5 MyTracks の開発シナリオ

図 9 は，前節までの説明に基づき，MyTracks の区画化を概要図として示したものである．今回は既存のアプリに対して提案方式を適用したが，実際には，まず，コンポーネント設計のフェーズにおいてプライバシポリシに従って図 9 のような区画化を行い，その上で，ドメイン間のメッセージパッシング（3.2 節）と位置情報の取得タイミング（4.3 節）に注意してコーディングを進めていくという開発フローとなる．

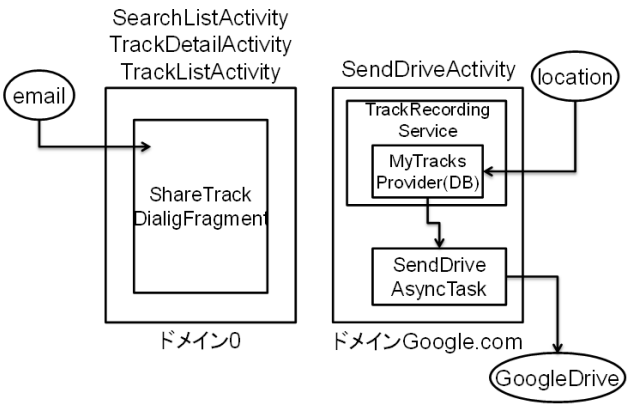

図 9. MyTracks の区画化の概要図

MyTracks は，利用するプライバシ情報は 2 つだけであり，その内，外部に送信されるプライバシ情報は 1 つのみであったため，区画化の適用が比較的容易なアプリケーションであった．利用するプライバシ情報や外部送信先を多数含むアプリや，規模が大きいアプリにおいては，ドメインの定義やドメイン間のデータの受け渡しが複雑になるため，開発者の負担が大きくなることが懸念される．そのようなアプリに対する提案方式の適用可能性についても早急に調

査を行いたい.

7. 考察

7.1 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る．このため，それぞれのドメイ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．ドメイ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，開発者への負担が増すことになる．この問題については，開発者に専用のビルドツールや開発プラグインを提供するなどの対応が考えられる．具体的な方策については今後の検討を進めて行く．

7.2 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．

7.3 提案方式の実現可能性と評価

本稿では, MyTracks という 1 つのアプリに対して提案方式の適用を試み，提案方式の実現可能性を評価した．しかし，全てのアプリに対して提案方式を適用できるかどうかについてはまだ検証できていない．また，4.5 節や 5 章に記したような制約や制限が存在する．提案方式の改良および拡張，そして，その評価方法について今後検討を進めていく必要がある．

8. おわりに

本稿では，プログラムの内部構造を「ドメイン」と呼ばれる区画で分離することによって，プログラム内でのプライバシ情報の伝搬範囲を限定する仕組みを提案した．提案方式は，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ことができる．Android アプリを対象に，提案方式による具体的なアプリ開発について検討した．

現時点では，まだ特定のアプリを対象とした適用評価しか行っていない．今後は複数のアプリを対象として提案方式の適用評価・検証を進めていく必要がある．また，開発者の負担を更に軽減する方法の検討，静的検査ツールの実装・評価なども今後の課題として挙げられる．

開発者の負担軽減については，各コンポーネントの適切なドメインへの自動分類や，ドメイ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，開発者にビルドツールを提供することによって，アプリのビルド完了と同時にアプリケーションプライバシポリシとの整合性検証署名が与えられるような仕組みの導入を検討している．

謝辞

本研究を進めるにあたり KDDI 竹森敬祐様に貴重なアドバイスを賜りました．ここに感謝の意を表します．

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) 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
6) Adrienne Porter Felt, et.al.: “How to Ask For Permission”, In Proc. USENIX Workshop on Hot Topics in Security, 2012
7) 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
8) 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, 2013-CSEC-62,2013.07
9) 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
10) 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)
11) Android の重要な機能，インテント: http://thinkit.co.jp/article/921/1 (2014.04.11)
12) MyTracks for Android
https://code.google.com/p/mytracks/
13) 小林信也 鈴木富明 可児潤也 川端秀明 竹森敬祐 西垣正勝:” 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”，情報処理学会研究報告,2014-CSEC-65,2014.05