Android端末における 情報フローを単位とする出力制御機構の提案

梶原 直也[1]　堀 良彰[2]　櫻井 幸一[2]

概要：Android 端末の普及に伴って，端末に保存されたプライバシ情報を漏洩させるアプリケーションが増加している．こうしたアプリケーションへの対策として，AndroidOS にはパーミッション機構というアクセス許可機構が存在する．しかし，パーミッション機構には，アプリケーションによるプライバシ情報の使用方法を制御できないという問題が存在する．本研究では，情報フローを用いてプライバシ情報の取得から送信までの一連の流れを監視する新しい制御機構を提案する．また，この制御機構によって取得する情報の使用範囲まで制御できる新たなパーミッションが作成可能であることを示す．

キーワード：Android，セキュリティ，プライバシ保護，パーミッション，情報フロー追跡

Proposal of Output Control System with Information Flow in Android

Abstract: With the spread of Android devices, applications that let privacy information in terminals leak out are increasing. There is an access control system called “Permission” in AndroidOS as a countermeasure against such malware. However, it is difficult to confirm if information will be sent out from terminals or not from permissions. We propose a new control system monitoring privacy information using information flow tracking. Furthermore, we indicate that new permissions controlling range of information can be created by this control system.

Keywords: Android, Security, Privacy Protection, Permission, Information Flow Tracking

1. はじめに

Android 端末の普及に伴って，そのセキュリティが注目を浴びている．これは，ユーザに悪影響を及ぼすマルウェアと呼ばれるアプリケーション (以下，アプリとする) の増加が原因である．特に，近年ではユーザの同意を得ることなしにプライバシ情報を取得し，外部へ送信するアプリの数が増え，プライバシ情報の漏洩が問題となっている．

こうした情報漏洩を引き起こすマルウェアへの対策として，AndroidOS にはパーミッション機構というアクセス許可機構が存在する．パーミッション機構は，Android 端末における動作の中でも，特にシステムやセキュリティに関する重要度の高いものに関して制御を行っている．ユーザは，アプリのインストール時に各パーミッションを許可するという形で，アプリへ各動作を行う権限を与えることになる．そのため，アプリが要求するパーミッションを確認することで，ユーザはアプリの動作を把握することができると考えられている．しかし，取得した情報の使用範囲を制御するパーミッションは存在しないため，取得された情報が端末外へ送信されるかパーミッションから判断することは難しいという問題点が存在する．

本研究では，パーミッション機構が API 単位で Android の動作を制御し，パーミッションによって管理している点に着目した．取得した情報の使用範囲が不明瞭であるという問題は，ここから生じていると考えられる．この問題に対して，情報フローを用いてプライバシ情報の取得から送信までの一連の流れを監視する新しい制御機構を提案する．この制御機構によって取得する情報の使用範囲まで制御で

[1] 九州大学工学部電気情報工学科
Department of Electrical Engineering and Computer Science, Kyushu University
[2] 九州大学システム情報科学府情報学専攻
Department of Imformatics, Kyushu University

きる新たなパーミッションが作成可能であることを示す．また，Android マーケットやアプリの現状を踏まえた上でこの制御機構をどのように機能させるべきか考察する．

2. Android のセキュリティ

2.1 Android 端末におけるプライバシ情報

本研究は，Android 端末に保存されるプライバシ情報の保護を目的とする．Android における代表的なプライバシ情報を表 1 にまとめる．

プライバシ情報
端末識別番号 (IMEI)
電話番号
SIM シリアル番号
加入者識別番号 (IMSI)
位置情報
連絡先データ
Gmail アドレス等のアカウント

表 1 Android におけるプライバシ情報

2.2 パーミッション機構

パーミッション機構は，AndroidOS において API やリソースの使用を制御するためのアクセス許可機構である．開発者は，作成したアプリに対して特定の動作を実装する場合，対応するパーミッションをマニフェストファイル内において宣言する必要がある．例えば，連絡先データをアプリに取得させる場合，“READ_CONTACTS” というパーミッションをマニフェストファイル内に記述する必要がある．もし，必要とされるパーミッションをマニフェストファイル内で記述しなかった場合，エラーが発生しアプリは正常に動作しない．そのため，パーミッションの明記は開発者に義務付けられ，ユーザはアプリのインストール時に，そのアプリが要求するパーミッションの一覧を確認することができる．パーミッション確認画面を図 1 に示す．

以上の流れが確立されているため，パーミッション一覧から得られる情報を判断材料として，ユーザはアプリをインストールするかどうか任意に選択できる．例えば，表 2 に示したような個人情報の取得に関するパーミッションと，インターネット通信を行うために必要な “INTERNET” パーミッションの組み合わせを持つアプリは，情報の外部への送信を行うことができるので注意する必要がある．

2.3 パーミッション機構の問題点

2.3.1 ユーザに対する不親切さ

はじめに，ユーザに対するパーミッションの分かりづらさが挙げられる．アプリのインストール画面において，ユーザが確認できる個々のパーミッションについての説明文は非常に簡素である．さらに，パーミッションの種類は非常に多く，それらの組み合わせパターンは膨大な数になる．そのため，パーミッションの一覧を確認するだけで，アプリの動作をユーザが正確に把握することは難しい．また，アプリをインストールする場合，そのアプリが要求する全てのパーミッションに対して，ユーザは承認を与える必要がある．これによって，あるパーミッションの組み合わせにユーザが脅威を感じたとしても，アプリをインストールするためには全てのパーミッションを承認する必要がある．例えば，前述したような個人情報の取得（例：“READ_CONTACTS”）とインターネットアクセス (“INTERNET”) というパーミッションの組み合わせをアプリが要求していた場合，情報の端末外への流出を避けるために，“READ_CONTACTS” パーミッションだけを承認することは不可能である．

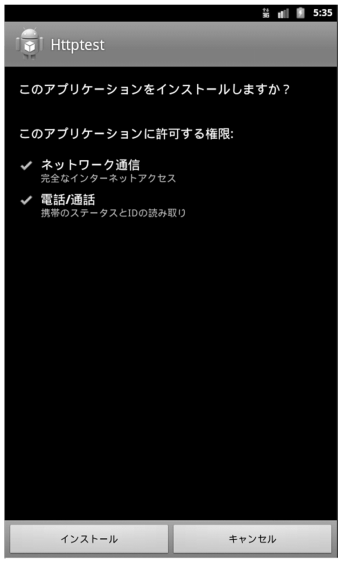

図 1 パーミッションの確認画面

パーミッション名	制御する動作
READ_CONTACTS	連絡先データの読み取り
READ_PHONE_STATE	電話のステータスと ID の読み取り
ACCESS_FINE_LOCATION	詳細な位置情報の使用
ACCESS_CORSE_LOCATION	おおよその位置情報の使用
READ_SMS	SMS の読み取り
READ_HISTORY_BOOKMARKS	ブラウザの履歴とブックマークの読み取り
READ_CALENDAR	カレンダーデータの読み取り

表 2 プライバシ情報の取得を制御するパーミッションの例 [1]

2.3.2 プライバシ情報保護に関する問題

次に，プライバシ情報の保護に関する問題が存在する．表 2 に示したように，既存のパーミッション機構において，プライバシ情報に関するパーミッションは複数存在する．しかし，これらのパーミッションは全てプライバシ情報の取得動作のみを制御するパーミッションであって，取得した情報の使用方法は制御しない．そのため，これらのパーミッションを確認したとしても，情報の用途や外部への出力の有無をユーザが把握することは不可能である．例えば，連絡先データを取得するための “READ_CONTACTS”

パーミッションとインターネットアクセスのための “INTERNET”パーミッションをアプリが要求していた場合，アプリが行う挙動の代表的な二つの例を，図 2 に示す．

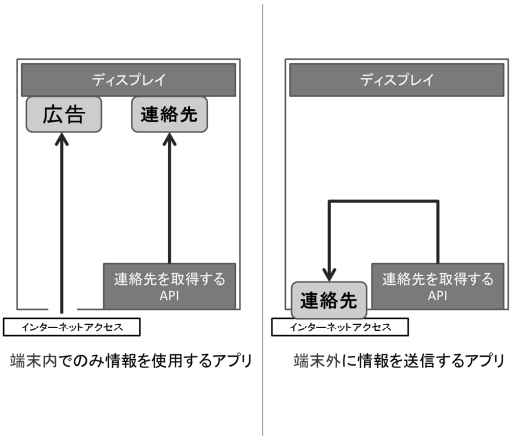

図 2 セキュリティ機構として
パーミッション機構が機能しない一例

図 2 において, 左に示すアプリは, “READ_CONTACTS” パーミッションによって取得した連絡先データを端末ディスプレイに表示し, それと並行して “INTERNET”パーミッションによって確立したインターネットアクセスを利用して外部から取得した広告を端末ディスプレイに表示する．それに対して，右に示すアプリは，“READ_CONTACTS” パーミッションによって取得した連絡先データを，“INTERNET”パーミッションによって確立したインターネットアクセスを通じて端末外部へと送信している．言い換えると，図中左のアプリは端末内でのみプライバシ情報を使用するアプリであるのに対して，図中右のアプリは端末外へプライバシ情報を送信するアプリだということができる．プライバシ情報保護の観点からみると，これら二つの動作は明確に区別されるべきであるが，二つのアプリが要求するパーミッションの組み合わせが全く同一であるため，パーミッションからこれらの動作を区別することは不可能である．

3. 既存研究

3.1 情報フロー追跡を用いた情報の送信動作検知

Android のセキュリティに関する問題点の中でも，アプリによる情報の送信状況を把握できないことは大きな課題であった．TaintDroid[2] は，この問題を解決するために，汚染追跡 (taint tracking) と呼ばれる情報フロー追跡技術を Android 端末において実装した研究である．

汚染追跡とは，追跡対象となる情報に対してタグ付けを行い，そのタグによって情報フロー追跡を行う手法である．TaintDroid は，端末内におけるプライバシ情報を取得された時点から動的に追跡し，アプリが情報を外部へ送信した場合，ユーザへ通知する．このシステムによって記録されたアプリによる情報送信を解析することで，マーケットで配布されているアプリのプライバシ情報利用状況が報告されている．しかし，TaintDroid にアプリ内の情報のやり取りにおいて，制御フロー構文が用いられた場合，追跡が失敗するという問題点が存在する．

3.2 アプリのインストールシステム改造による選択可能パーミッションの実現

Apex[3] は，パーミッション機構の改造を行った研究である．

パーミッション機構には，アプリが要求する全てのパーミッションを承認しなければ，アプリをインストールすることができないという問題点が存在した．そのため，開発者が設定したアプリに与える権限を，ユーザ側で制限することは不可能であった．Apex は，Poly という名の Android アプリインストーラを作成して，アプリが要求しているパーミッション各々に対する個別の承認を可能とすることで，この問題を解決した研究である．また，時刻やデータへのアクセス回数等の条件に応じた，パーミッション承認の動的な削除機能も実装している．

3.3 アプリ実行ファイルのデコンパイルによる静的解析手法

ded[4] は，Android アプリに対する静的解析の易化を目的とした研究である．

通常，マーケットで公開されているアプリのソースコードを入手することは不可能であるため，ソースコードに対する静的解析は困難だった．ded は，アプリの実行ファイルである dex ファイルの java コードへのデコンパイルを実現することによって，この問題を解決した研究である．また，Android マーケットにおいて公開されているアプリ 1,100 個のアプリを実際にデコンパイルし，ソースコードを解析することによって，Android アプリに関する詳細な報告を行った．

4. 提案手法

4.1 概要

Android のパーミッション機構には，プライバシ情報の使用範囲をユーザが把握することが難しいという問題点が存在した．この問題への対策として，本研究では，情報フロー追跡を用いた新たな制御機構を提案する．これは，アプリによって取得されるプライバシ情報を情報フロー追跡によって監視することで，取得された情報が端末内でどのように使用されるかを制御する手法である．また，この制御機構によってプライバシ情報の使用範囲を制御する新たなポリシーが作成できると示す．

4.2 方針

既存のパーミッション機構における大きな問題は，プライバシ情報の出力の有無をパーミッションから確認することが不可能な点である．ユーザは，アプリがプライバシ情報を取得する権限を保有することはパーミッションから確認できるが，その情報をどのように使用するかは確認できない．そのため，既存のパーミッションはプライバシーポリシーとして機能しているとは言い難く，ユーザがパーミッションを承認してアプリをインストールしたとしても，その行動をユーザによるプライバシ情報取扱いへの同意とみなすことはできない．

本研究では，プライバシ情報に対する挙動の中でも，特に，ユーザの同意を得ること無しにプライバシ情報を外部へ出力することが問題であると考えている．したがって，提案手法は，プライバシ情報がアプリによって出力されるかどうか，ユーザが正確に把握できるシステムの構築を目的とした．この目的のために，情報フロー追跡を用いた新たな制御機構について提案する．

4.3 提案方式

従来，パーミッションの多くは，動作へのアクセス制御を API 単位で行なっている．例えば，端末識別番号の取得という動作は，“READ_PHONE_STATE” というパーミッションによって，“getDeviceId()” という API 単位で制御されている．そのため，プライバシ情報の取得動作を制御するパーミッションは存在するものの，取得された情報の使用方法を制御するパーミッションは存在しなかった．したがって，ユーザはアプリの情報使用方法について，マーケットにおける開発者の説明等から判断するしかない．しかし，開発者に悪意がある場合，情報の使用方法について虚偽の説明を行うことは容易である．また，開発者に悪意がなかったとしても，アプリの設計ミスや広告モジュールの動作を開発者が把握してなかった場合において，開発者の予期せぬプライバシ情報の使用が発生する可能性がある．以上のように，取得したプライバシ情報を，ユーザの同意無しにアプリが送信することを，現状のシステムにおいて防ぐことは難しい．この問題を解決するために，プライバシ情報の取得から出力までを監視し，それらの動作を制御する新たな機構が必要である．

上記の問題点改善のために，本研究では，情報フロー追跡によるプライバシ情報の監視を行う新たな制御機構を提案する．この制御機構によって，現パーミッション機構において使用されている情報の取得動作を制御するパーミッションを，

- 取得したプライバシ情報の出力を許可する
- 取得したプライバシ情報の出力を許可しない

という二つのパーミッションへ細分化する．これにより，プライバシ情報の使用範囲をユーザに対して明確に通知することが可能となる．

4.3.1 情報の追跡範囲

この場合，プライバシ情報に対する追跡の範囲をどこに設定するかは重要な要素である．本研究では，プライバシ情報漏洩の対策を目的としているため，プライバシ情報の漏洩原因となりうる情報出力に着目した．そのため，以下の三つの情報出力経路に対して監視を行う必要がある．

(1) ネットワーク通信
(2) プロセス間通信
(3) 外部ストレージへの書き込み

ネットワーク通信

プライバシ情報の出力経路として最も用いられるのはネットワーク通信である．アプリによるサービスへの利用や広告モジュールによる利用等，その使用方法も多岐に渡る．特に，モバイル端末は常時ネットワークに接続されている場合が多いため，ネットワーク通信においてプライバシ情報がどのように出力されるのか，監視を行う必要がある．

プロセス間通信

Android がもつ特色の一つに，アプリ間の連携における自由度の高さがある．ここで注意しておくべきことは，アプリ間での機能の連携やデータのやり取りを行う場合には，プロセス間通信が用いられる点である．これは，Android アプリがそれぞれ独立したプロセス上で動作しており，端末内の各アプリが直接通信を行うことは不可能なためである．. しかし，プロセス間通信を利用したアプリの連携は悪用される可能性がある [5], [6]．プロセス間通信を利用した代表的な攻撃は，脆弱性のあるアプリを踏み台にすることで，本来必要であるパーミッション無しに，悪意のあるアプリが特定動作を実行できるというものだった．過去には，実際に「設定」アプリに脆弱性が存在したため，「設定」アプリを踏み台にすることで，権限を持たないアプリでも端末の設定を変更する事が可能であった．同様に，もしプライバシ情報を取り扱うアプリに同様の脆弱性が存在した場合，プロセス間通信を用いることで，悪意のあるアプリは，必要なパーミッションなしにプライバシ情報を取得できる．以上より，プロセス間通信によるプライバシ情報の出力に対しても監視を行う必要がある．

外部ストレージへの書き込み

Android 端末の多くは，SD カードのような外部ストレージを利用することができる．ところが，Android4.0 までは外部ストレージへ保存されているデータの読み取り動作については，パーミッションで制御されていなかった．Android4.1 において，外部ストレージのデータ読み取り権限を管理する “READ_EXTERNAL_STORAGE” パーミッションが現れたが，現時点では実際に使用はされていない．また，このパーミッションの実装にかかわらず，プライバシに関わる重要な情報を外部ストレージに保存する行為に

は，一定のリスクが存在する．したがって，外部ストレージへのプライバシ情報の書き込みについても，監視を行うべきである．

4.4 実装方法

4.4.1 実装ポリシー

提案手法の目的は，大きく分けて 2 つある．1 つ目はプライバシ情報の使用範囲制御，2 つ目はその使用範囲のユーザへの通知である．

プライバシ情報の使用範囲制御は，近年問題となっている Android 端末内のプライバシ情報漏洩問題への対策を目的としている．現在，様々な Android マルウェアが出現しているが，そのうち日本で問題となっているものの多くは，ユーザのセキュリティ意識の低さを前提として作成されている．例えば，端末内の連絡先データをユーザの同意無しに取得し，外部へ送信してしまうアプリが 2012 年に多く発見された [7]．しかし，このようなマルウェアの多くは，端末内で取得したプライバシ情報を単純に外部サーバへ送信するだけのものであった．提案手法によって，こういったマルウェアによるプライバシ情報の漏洩を防ぐ．また，現在の Android アプリは多様化が進む一方であり，アプリの情報用途に関しても悪用かどうかの機械的な判定が難しくなってきている．そのため，アプリによるプライバシ情報の使用状況はユーザに対して明確に通知されるべきである．

提案手法では，以上の目的を達成するために，情報フロー追跡技術の一種である汚染追跡と，パーミッション機構に関連するユーザインタフェースの連携を提案する．

4.4.2 汚染追跡による情報の使用範囲監視

端末内におけるプライバシ情報の監視には汚染追跡 [2] と呼ばれる既存手法を用いる．これは，端末内で取得されたプライバシ情報にタグを付加して追跡を行う手法である．TaintDroid[2] が Android において実現した汚染追跡は，4.3.1 節で述べた三つの情報出力経路に対しても追跡を行なっている．

提案手法における情報の監視手順を以下に示す．

(1) 端末内で取得されたプライバシ情報を格納する変数に対して，タグ付けを行う．
(2) 代入や演算結果，関数の引数，返り値など，タグ付けされた変数に関わった変数に対してそれぞれタグは伝播していく．
(3) タグ付けが行われている変数が出力されようとした場合，その動作をタグによって検知する．
(4) 出力が完了される前にその動作を中断させることで，追跡対象となっているプライバシ情報が外部へ出力されることを防ぐ．

以上の手順によって，本研究で提案する「取得したプライバシ情報の出力を許可しない」というパーミッションを実現することが可能である．また，上記手順における (4) 出力の中断の中でも，特に，ネットワークアクセスを介した通信の中断に関しては，TaintDroid を発展させた研究 [8] において既に実現されている．

4.4.3 アプリインストーラの改造によるユーザインタフェースの実装

提案手法における目的の一つは，ユーザに対してアプリが使用するプライバシ情報の使用範囲を通知させることである．そのため，ユーザインタフェースの実装は必須である．今回は，既存のアプリインストール画面を改造することによって，ユーザインタフェースの実装を目指す．

4.5 パーミッションの設定方法

提案手法では，プライバシ情報の使用範囲まで制御するパーミッションによって，プライバシ情報の使用範囲をユーザへ通知することができると示した．この場合，パーミッションの設定方法として，以下の二つの方法が考えられる．

(1) アプリ開発者によるパーミッション設定
(2) ユーザによるパーミッション選択

4.5.1 アプリ開発者によるパーミッション設定

提案手法における新たなパーミッションの設定方法として，まず考えられるのがアプリ開発者によるパーミッションの設定である．既存のパーミッションを用いる場合と同様に，プライバシ情報の使用範囲を制御するパーミッションを，開発者がマニフェストファイル内に記述することで設定を行う．具体的には，アプリに取得させたプライバシ情報を外部へ出力させる場合は「出力を許可する」パーミッションを，出力させない場合は「出力を許可しない」パーミッションをマニフェストファイル内で記述する．この場合の利点と欠点を以下にまとめた．

利点

(1) 開発者自身がパーミッションという形でポリシーを作成できる
(2) 広告モジュールとの親和性が高い
(3) ユーザに対して要求される操作が少ない

アプリ開発者がパーミッションを設定する場合の最大の利点は，パーミッションを開発者によるプライバシーポリシーとして運用できる点にある．既存のパーミッションでは，プライバシ情報の取り扱いまで制御できていなかったため，プライバシーポリシーとしては不十分であったが，本研究で提案するパーミッションは，プライバシ情報の取り扱いまで制御を行うため，開発者発信のプライバシーポリシーとして十分実用可能である．

また，現在無料アプリの多くには広告モジュールが含まれているが，これらの広告がプライバシ情報を取得し外部へ送信している事実が報告されている [4]．このようなプライバシ情報の広告への使用はユーザへ通知されない場合

図 3 インストール画面のイメージ

も多く，適切にプライバシ情報が取り扱われているとは言いがたかった．しかし，提案手法によるパーミッションを開発者が設定することで，プライバシ情報の広告による利用をユーザへ通知し，より適切な形で広告モジュールを運用することが可能となる．

欠点

(1) 開発者がアプリやモジュールの挙動を正確に把握する必要がある
(2) ユーザがパーミッションを確認するとは限らない
(3) 現在使用されているアプリには適用できない

開発者がパーミッションを設定する場合，アプリやモジュールの動作を正確に把握する必要がある．例えば，アプリがプライバシ情報の出力を行なっているにもかかわらず，開発者が「出力を許可しない」パーミッションをアプリに付加してしまった場合，アプリが正常に動作を行えなくなってしまう．反対に，アプリがプライバシ情報の出力を行わないにもかかわらず，開発者が「出力を許可する」パーミッションをアプリに付加することもありえる．この場合，アプリの実際の動作には必要ない過剰な権限を与えてしまうことになり，情報使用ポリシーとしてパーミッションが機能しなくなるという問題がある．

また，ユーザのセキュリティ意識が低い場合，アプリのインストール時にパーミッションを確認しない可能性がある．そのため，アプリ開発者が正しくポリシーを制定できたとしても，それが正常に機能するとは限らない．

4.5.2 ユーザによるパーミッション選択

アプリ開発者が予めプライバシ情報の使用範囲をパーミッションによって設定せずに，ユーザ自身がパーミッションを選択することによって，プライバシ情報の使用範囲を制御する事が可能である．今回は，図 3 に示すように，アプリのインストール時に表示されるパーミッション確認画面において，ユーザがプライバシ情報の使用範囲を指定するシステムについて，利点と欠点を考察する．

利点

(1) 悪意のあるアプリに対して有効
(2) 現在使用されているアプリに対して適用出来る

ユーザによるパーミッション選択を実装した場合の最大の利点は，悪意のあるアプリに対して有効だという点である．この場合，悪意のあるアプリとして，ユーザの同意を得ること無しに，意図的に端末内のプライバシ情報を漏洩させるアプリを想定している．ユーザ自身がプライバシ情報の使用範囲を制限することによって，それらの悪意あるアプリによる情報の漏洩を防ぐことが可能である．

また，パーミッションの選択をユーザが行う場合，アプリ開発者が，アプリの作成手順において，現在の手順と比較して追加で行う動作は存在しない．そのため，現在マーケットなどで公開されている既存のアプリに対しても，プライバシ情報の制御が行えるという利点が存在する．

欠点

(1) 現状のビジネスモデルとの親和性が低い
(2) ユーザに対してある程度の情報リテラシーが要求される

先程も述べたように，無料アプリには，様々な広告モジュールが組み込まれており，その中にはユーザのプライバシ情報を広告に利用しているものが存在する．そのため，プライバシ情報の出力をユーザが制限した場合，広告モジュールは正常な動作を行えなくなる．このことによって，広告によるビジネスモデルが機能しなくなるという問題が存在する．

もう一つの欠点として，ユーザに対してプライバシ情報に関するリテラシーが要求されることが挙げられる．ユーザ自身がプライバシ情報の使用範囲を選択するという手法は，ユーザが正しい知識のもとに判断を行えるという前提によって成り立つため，その判断を正しく行えない場合には，提案手法は機能しない．

5. 課題

今後の課題は大きく分けて二つ挙げられる．

一つ目は，提案手法の実装である．提案手法は，既存手法である汚染追跡を利用したプライバシ情報の使用範囲制御と，ユーザインタフェースを連携させることによって実現される．Android において汚染追跡を用いた TaintDroid[2] では，タグによるプライバシ情報の監視は行なっていたが，情報を取得したアプリごとにタグを管理し，その動向を制御する機能は実装されていない．今後は，それらの機能を実装し，ユーザインタフェースとの連携を実現する必要がある．

二つ目は，汚染追跡による追跡限界の解決である．提案手法で使用する汚染追跡技術には，制御フローの使用により追跡に用いる汚染タグを取り外すことができるという問題点 [9] が存在する．図 4 に例を示す．

```
switch (x) {
        case ‘a’ : y=‘a’;
                   break;
        case ‘b’ : y=‘b’;
                   break;
}
```

図 4 制御フローによって汚染追跡が失敗する例 [9]

これは，変数 x に格納されている値を変数 y にそのまま代入するだけのコードであるが，この時，汚染追跡による監視中でも，変数 x に付加されている汚染タグが変数 y に移ることはない．そのため，マルウェア作成者が上記のような制御フローをアプリ中に意図的に組み込むことによって，タグ付けによる追跡を振り切ることができる．この問題への対策として，アプリの静的解析による制御フロー構文の発見が有効である．しかし，提案手法は Android 端末上での実装を行うため，CPU に対する負荷の大きい静的解析との連携は難しい．そこで，今後の課題として，Android 端末上で実行可能な特定構文の発見手法等の対策を考案する必要がある．

6. おわりに

6.1 まとめ

本研究では，Android 端末内のプライバシ情報を保護するために，既存のパーミッション機構に代わる新たな制御機構を提案した．この制御機構は，情報フロー追跡を用いて，端末内で取得されたプライバシ情報を監視し，情報の使用範囲を制御するというものである．この手法を用いることによって，既存のパーミッション機構ではユーザが知ることの出来なかった情報の使用範囲を明らかにし，ユーザがアプリの挙動をより正確に把握できるようになる．

6.2 今後の課題

今後の課題は大きく分けて二つ存在する．

一つ目は，提案手法の実装である．本研究において提案されたプライバシ情報の取得から出力までを監視する制御機構は，情報フロー追跡とインタフェース実装の組み合わせによって実現される．また，提案手法の実装後，マーケットで実際に公開されているアプリを対象とした実験，評価を行う必要がある．マーケットで公開されているアプリのプライバシ情報使用用途や使用方法は多岐にわたるため，特に，汚染追跡を実装した端末がどの程度の負荷を受けるか，プライバシ情報の追跡はどの程度の精度で行えるか，といった点を実験することが重要である．

二つ目は，汚染追跡技術の限界解決である．タグによって情報を追跡する汚染追跡技術には，制御フロー構文による情報の受け渡しが行われると追跡が失敗するという問題点が存在する．今後の課題として，この問題を解決する必要がある．

謝辞 本研究において，有益なコメントを下さった九州大学の西出隆志助教に感謝の意を表する．本研究の一部は，JSPS 科研費 23300027 の助成を受けたものである．また，本研究の一部は（財）電気通信普及財団の研究助成によって行われた．本研究を進めるにあたり，有益な助言をいただいた株式会社 KDDI 研究所窪田歩氏，同磯原隆将氏に感謝する．

参考文献

[1] タオソフトウェア株式会社『Android Security 安全なアプリケーションを作成するために』インプレスジャパン，2012
[2] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. , “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, ”In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), October, 2010. Vancouver, BC.
[3] Mohammad Nauman, Sohail Khan, Xinwen Zhang, “Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints, ”In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, April, 2010, 328-332
[4] William Enck, Damien Octeau, Patrick McDaniel, Swarat Chaudhuri, “A Study of Android Application Security, ”In Proceedings of the 20th USENIX Security Symposium, August, 2011. San Francisco, CA.
[5] William Enck, Machigar Ongtang, and Patrick McDaniel. “Understanding Android Security.” IEEE Security & Privacy Magazine, 7(1):50-57, January/February 2009.
[6] Adrienne Porter Felt, Helen J. Wang, Alexander Moshchuck, Steven Hanna, and Erika Chin. “Permission re-delegation: Attacks and defenses.” In 20th Usenix Security Symposium, San Fansisco, CA, August, 2011.
[7] “情報処理推進機構：情報セキュリティ：ウイルス・不正アクセス届出状況について（2012 年 8 月分）”(最終確認日:2013 年 1 月 30 日)，http://www.ipa.go.jp/security/txt/2012/09outline.html
[8] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, David Wetherall, “These Aren’t the Droids You’re Looking For: Retrofitting Android to Protect Data from Imperious Applications, ”In Proceedings of the 18th ACM Conference on Computer and Communications Security, October, 2011.
[9] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, Engin Kirda, “Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis, ”in Proceedings of the 14th ACM Conference on Computer and Communication Security, Alexandra, VA, October, 2007.