Privacy protection using masquerade pointer on Android OS

Vol.2012-CSEC-57 No.18
Vol.2012-IOT17 No.18
2012/5/11

Android OS におけるマスカレーディングポインタを用いたプライバシー保護

上松晴信[†] 可児潤也[†] 名坂康平[†] 川端秀明[††] 磯原隆将[††] 竹森敬祐[††] 西垣正勝[†††]

近年，スマートフォンのセキュリティが重要視されてきている．特に，情報漏洩，不正課金，ワンクリック詐欺などの問題が多発し，社会問題となっている．本稿では，Android OS 内にセキュリティマネージャと呼ばれる機構を設け，マスカレーディングポインタを用いて機密情報管理を行うことでこれらの問題を解決する方式を提案する．アプリに機密情報に対する参照ポインタのみを渡し，機密情報に関してはセキュリティマネージャ側で管理および処理を行うことで安全な機密情報の取り扱いが可能になる．

Privacy protection using masquerade pointer on Android OS

HARUNOBU AGEMATSU　JUNYA KANI　KOHEI NASAKA
HIDEAKI KAWABATA　TAKAMASA ISOHARA　KEISUKE TAKEMORI
MASAKATSU NISHIGAKI

Security of smart-phone is considered as important. Especially the number of leakage of privacy information, incorrect billing, and one-click billing fraud has been increasing recently, and they cause many problems. This paper proposes a new security measure to protect privacy information; “security manager” and “masquerade pointer”. The security manager returns the reference pointer for the privacy information, instead of the privacy information itself, when any Android application sends a request for it to the OS. By doing so, the privacy information is masqueraded in Android applications, and thus applications are prevented from abusing the information.

1. はじめに

近年，Android OS が搭載された Android フォンが爆発的に普及している．Android フォンの特徴として，Android フォンを持つユーザは Google Play Store [1]にアクセスし，アプリケーションソフトウェア（以下，アプリ）を自由にダウンロードしてインストールできることが挙げられる．また，Android アプリの開発環境は無償で提供されているため，誰でも自由にアプリを作成し，Google Play Store にアップロードすることができる．

Google Play Store のような有名なマーケットは全てのアプリケーションをチェックし,不正アプリを削除している．しかし Android OS の場合, ネット上には信用できないマーケットも存在しており，そこではトロイの木馬と呼ばれる正規のアプリを装った不正アプリ（ワンクリックウェア [2], Geimini [3]等）が紛れ込み，様々な被害をもたらしている．これらの不正アプリはインストールされると，フォアグラウンドでは正規のアプリと同様に振舞いながら，バックグランドでユーザの個人情報（電話番号，端末番号，位置情報，SMS メッセージ，通話記録など）を取得しそれらを外部のサーバへ送信することで，ユーザに気づかれることなく情報漏洩を引き起こす．

不正アプリの被害を減らすためのアプローチとして，ユーザ側で対策を行う方法とアプリ提供側で対策を行う方法が考えられる．

ユーザ側での対策としては，アプリインストール時に表示されるパーミッションを各ユーザが注意深く確認し不正アプリかどうかの判断を行うことや，アンチウイルスソフトを活用するなどの方法が考えられる．しかし，Android フォンを利用しているユーザのセキュリティに対する意識や知識は様々であり，ユーザ側で完璧な対策を行うことはきわめて困難である．また，文献[4]では，セキュリティ対策ソフトでの対策では限界があることが述べられている．

アプリ提供側での対策としては，提供するアプリをあらかじめ検査しておき，安全であると判断されたアプリのみをマーケットにて提供する方法が考えられる[5][6]. アプリ提供側で対策を行うことにより，ユーザのセキュリティ意識などに依存せず，すべてのユーザに対して安全なアプリのみを提供することが可能となる．しかしながら不正アプリは多種多様で膨大であり，すべての不正アプリを完全に漏れなく検出することは現実的には困難である．

そこで本稿では，Android フォンの中の機密情報をより安全に扱う方法を提案する．具体的には OS 内に機密情報を管理する「セキュリティマネージャ」という機構を設ける．OS が管理している機密情報をアプリが利用したい場合，(i)　アプリは OS に機密情報の取得要求を送り，これ

[†]静岡大学大学院情報学研究科，〒432-8011 浜松市中区城北 3-5-1,
Graduate school of Informatics, Shizuoka University, 3-5-1 Johoku, Naka, Hamamatsu, 432-8011 Japan
[††]株式会社 KDDI 研究所，〒356-8502 埼玉県ふじみ野市大原 2-1-15
KDDI R&D Laboratories, Inc. 2-1-15 Ohara, Fujimino, Saitama, 356-8502 JAPAN
[†††]静岡大学創造科学技術大学院，〒432-8011 浜松市中区城北 3-5-1,
Graduate School of Science and Technology, Shizuoka University, 3-5-1 Johoku, Naka, Hamamatsu, 432-8011 Japan

に対して（ii） OS が当該秘密情報をアプリに返送する．セキュリティマネージャは，（ii）の時点でアプリと OS の間に介在し，アプリに機密情報そのものではなく，機密情報を指し示す参照ポインタを返す．

アプリが機密情報を出力する際には，セキュリティマネージャが参照ポインタを機密情報に復元する．機密情報の出力先が自端末内の OS 管理下リソースである場合は，参照ポインタは自動的に当該機密情報に変換される．機密情報が自端末 OS 外に出力される場合は，その操作を行う前にユーザに確認が求められる．参照ポインタは機密情報をマスクする役割を果たすため，この仕組みを「マスカレーディングポインタ」と名付ける．

この機構によって，不正なアプリが機密情報を読み取るパーミッションを宣言し，ユーザがインストール時にこれを承認したとしても，アプリが取得できるのは参照ポインタのみとなり，不正アプリによる機密情報の漏洩を防ぐことができる．正規のアプリが機密情報を操作する場合には，セキュリティマネージャによって参照ポインタは真の機密情報に自動変換されるので，正規アプリは正常にその動作を完了することができる．

2. 関連研究

Enck らは，IMEI（International Mobile Equipment Identity や IMSI（International Mobile Subscriber Identity）等の機密情報を色付け（taint）し，この色付き情報の流れを捕捉することによって情報漏洩をリアルタイムに検知する TaintDroid という仕組みを提案している[7]. TaintDroid を実装して評価も行っており，そのオーバヘッドは最大 29%程度であったと報告している．

Chin らは，Android のインテントを用いたアプリ間連携の脆弱性について述べている[8]．具体的には，メッセージパッシングの途中でインテントの内容がインターセプトされ盗聴，改ざんされる可能性があることを指摘している．実際に 100 個のアプリケーションに対して調査を行い，それらのアプリが実際に脆弱性を持っていたことを明らかにしている．

川端らは Android における情報へのアクセス制御機構（Api Manager）を提案している[9]．API へのフックと Android フレームワークの拡張を行うことで，アプリによる API コールに対してセキュリティポリシを強制的に適用する．ポリシを参照して API コールをコントロールする時間を計測し，Api Manager 実装時のオーバヘッドが許容範囲内で抑えられていることを確かめている．

松戸らは Android アプリをインストールする際のセキュリティ助言システムを提案している[10]．パーミッションに基づきアプリの危険性の判断支援を行うユーザインタフェースを実装している．アンケートによるユーザ評価を行い，提案方式を許容する意見が 9 割を超えていたことを確かめている．

3. Android OS

Android OS で実装されているセキュリティモデルを 3.1 節で述べたあとに，その問題点を 3.2 節で述べる．

3.1 セキュリティモデル

Android OS には図 1 に示したように Dalvik と呼ばれる仮想マシンが搭載されており，全てのアプリはこのサンドボックス上で実行される．それぞれのアプリには異なるユーザ ID が割り当てられ，別々のメモリ空間で実行される．アプリが他のアプリと連携する際には，インテントと呼ばれるメッセージパッシング機能を用いる．また，アプリが OS 内の情報（端末の機密情報など）にアクセスするためには，アプリのインストール時にパーミッション（図 2）を提示し，ユーザの承認を得る必要がある．

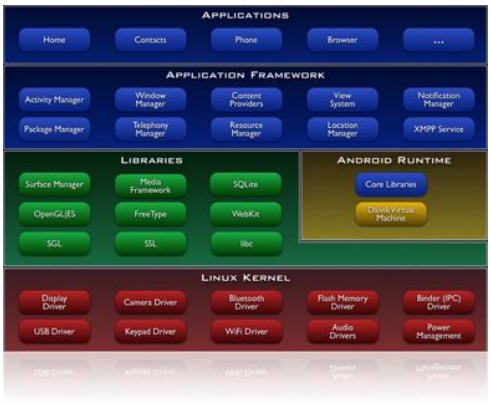

図 1. Android OS の構成[11]

図 2 アプリのパーミッション表示

アプリはパーミッションを得ることでさまざまな機能を提供する API（Application Program Interface）を使うことができる．API は図 1 の Application Framework 層に実装されている．例として，アプリが端末の機密情報を取得し表示するときの一連のフローを以下に示す（図 3）．

Step1） アプリが機密情報を取得する API をコールする．

Step2） OS が機密情報をアプリに返す.
Step3） アプリは受け取った機密情報を表示するために，当該機密情報を引数にして表示用 API をコールする.
Step4） OS が機密情報を端末画面に表示する.

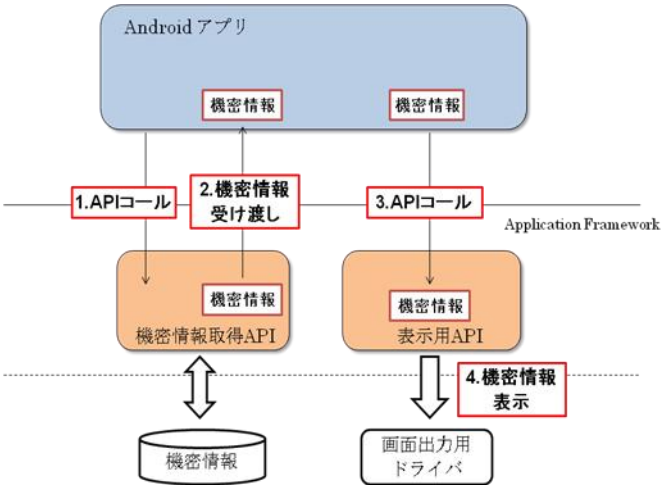

図 3. Android OS の機密情報管理

3.2 問題点

図 2 で示されるパーミッション情報は抽象的であり，この情報だけではユーザは具体的にアプリがどのような機能を有し，OS 内のどのリソースにアクセスするのか理解することが難しい．このため，一般的なユーザにとって，アプリインストール時にアプリが有する潜在脅威を認識してパーミッション可否を正しく判断することは困難であるという問題が指摘されている.

携帯電話には多くの個人情報が存在しており，不正アプリが一度端末にインストールされると様々な問題が引き起こされる．Android フォンにおいて起こり得る，具体的な脅威を以下に列挙する.

1. 端末内の識別情報（電話番号，SIM 番号，端末 ID 情報，国情報など）の漏洩
2. 端末内の通話履歴の漏洩
3. 端末内の電話帳データの漏洩
4. 端末の位置情報（GPS 情報）の漏洩
5. 端末内の送受信メール（SMS を含む）内容の漏洩
6. 端末内のカレンダー情報の漏洩
7. 端末からの不正電話発信
8. 端末からの不正メール（SMS を含む）送信
9. 端末からの不正インターネットアクセス

最近では，1 の端末情報の漏洩によって引き起こされるワンクリック詐欺[12][13]の事例が複数報告されている．携帯端末画面に自身の電話番号や IMSI 番号が記載された不正請求書が表示されるため，ユーザがその不正請求を誤信しやすい．また，スマートフォンの場合は一人一台の所有形態となるため，端末情報が不正者に漏洩してしまった場合にユーザが感じる精神的な不安は非常に大きい.

2～6 の情報はユーザのプライバシーに強く関係する情報である．7～9 については，1～6 の情報を不正者に送信する手段として用いられるだけでなく，通話料およびネットワークアクセス課金を不正に請求する手段，あるいは，不正者の踏み台として外部へのリモート攻撃を実行する手段として利用される.

4. 提案方式

本稿では，機密情報に対して参照ポインタ（マスカレーディングポインタ）を用意し，機密情報と参照ポインタを管理するセキュリティマネージャという機構を Android OS 内に導入する.

セキュリティマネージャは，アプリからの機密情報の取得要求に対し，機密情報の代わりに参照ポインタをアプリに返し，機密情報そのものは OS 内のテーブルにて格納する．アプリが当該機密情報を出力する際には，セキュリティマネージャが参照ポインタから機密情報への復元の可否を動的に判断する．機密情報の出力先が自端末内の OS 管理下リソースである場合（例えば機密情報を端末画面に表示する場合など）は，セキュリティマネージャは自動的に参照ポインタを真の機密情報に変換する．機密情報が自端末 OS 外に出力される場合（例えば機密情報を自端末内アプリに転送する場合，外部端末に送信する場合，自端末の SD カードへ保存する場合など）は，その操作を行う前にユーザに確認を求め，ユーザの承諾があったときのみ，セキュリティマネージャが参照ポインタを機密情報へと変換した上で OS がこれを出力する.

不正なアプリが機密情報を読み取るパーミッション（例えば READ_PHONE_STATE 等）を宣言し，ユーザがインストール時にこれを承認してしまったとしても，このアプリが OS から機密情報を実際に読み取ろうとした際には，セキュリティマネージャによってそれらの機密情報はすべて参照ポインタに置き換えられる．この結果，不正アプリに機密情報そのものが渡ることはなく，不正アプリによる機密情報の漏洩を防ぐことができる．正規のアプリが機密情報を操作する場合には，セキュリティマネージャが動的に参照ポインタを真の機密情報に自動変換するので，機密情報を扱う正規アプリは正常にその動作を完了することができる.

3.1 節および図 3 にてアプリが端末の機密情報を取得し表示する際のフローを示したが，提案方式を実装した場合には，このフローが以下のように変わる（図 4）.

Step1） アプリが機密情報を取得する API をコールする.
Step2） セキュリティマネージャは乱数等によって参照ポ

インタを生成し，参照ポインタと機密情報のペアをテーブルに格納する．

Step3） セキュリティマネージャは参照ポインタをアプリに返す．

Step4） アプリがこの機密情報を表示する場合は，参照ポインタを引数にして表示用 API をコールする．

Step5） セキュリティマネージャはテーブルを参照して，参照ポインタを対応する機密情報に変換した上で OS に渡す．

Step6） OS が機密情報を端末画面に表示する．

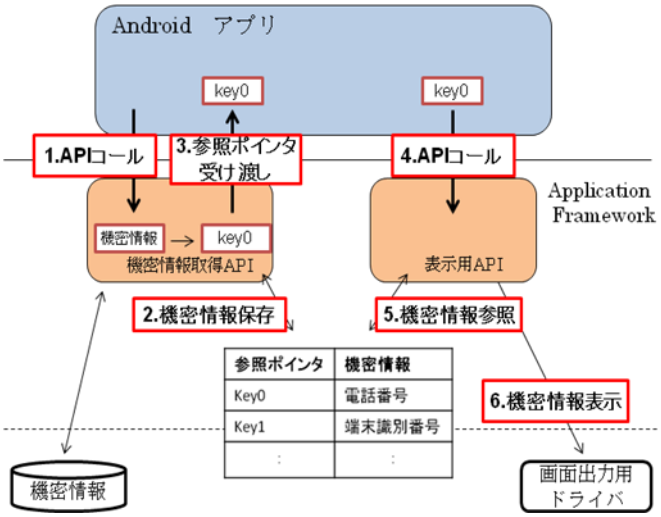

図 4. セキュリティマネージャによる機密情報管理

ここで，図 4 は機密情報を自端末の画面に表示する場合の例であるため，Step 5 においてユーザへの許諾確認は行われない．機密情報を OS 管理外へ出力する場合は，Step 4 および Step 6 にてコールされる API が SD 保存用の API に代わるとともに，Step 5 においてユーザに確認ダイアログが表示さることになる．

5. 実装

Android OS に手を加え，セキュリティマネージャのプロトタイプを実装した．OS のビルド環境は表 1 のとおりである．今回は，機密情報の端末画面への表示，および，SD カードへの書き込みを想定した．

図 4 における機密情報管理テーブルは Java の配列を用いて実装し，Step 2 の参照ポインタの生成（今回は key0, key1, ・・・をポインタの値とした）およびテーブルへの格納，Step 5 のテーブル参照による参照ポインタからの機密情報の復元についてはそれぞれ独自の API を作成した（表 2）．詳細については，ソースコードとその説明を付録 A に掲載したので参照されたい．

表 1. Android OS のビルド環境

OS	Ubuntu10.04(lucid)
Kernel version	カーネル 2.6.32-39-generic
memory	1.8G
CPU	Intel Core2 Duo CPU E6750 2.66GHz
Android version	Android2.3.5 GINGERBREAD
Kernel version	2.6.35.7-g3cc95e3

表 2. 作成したクラスと API

クラス	API	機能
SecurityManager	secretInsert	テーブルへの格納
SecurityManager	secretSearch	テーブル参照

Android OS の既存 API の内，表 3 に示した API をフックし，表 2 の自作 API を挿入することによって，図 4 のフローを実現する．今回は，機密情報取得 API を 5 つに絞って実装をしたが，他の情報取得 API に対しても同様の改造が可能である．

表 3. フックした API

図 4 との対応	クラス	API
機密情報取得 API	TelephonyManager	getLine1Number
		getDeviceId
		getSimCountryIso
		getSimOperator
		getSimSerialNumber
表示用 API	TextView	setText
SD 保存 API	BufferedWriter	Write

5.1 機密情報の端末画面への表示

端末の電話番号，デバイス ID，SIM の国名，SIM の国コード，SIM 番号を取得し，これを画面に表示するアプリを作成し，提案方式を実装した Android フォンにて実行した．

図 5 はアプリ側で保持している値を Android のログ機能（Logcat）によって出力した画面である．Message 欄に表示されている情報がアプリの所持している値である．電話番号，デバイス ID，SIM の国名，SIM の国コード，SIM 番号がそれぞれ参照ポインタ key0，key1，key2，key3，key4 に置き換わっている．つまり，アプリが取得したのは端末情報そのものではなく参照ポインタの値であることが分かる．

図 6 は，アプリがこれらの端末情報を端末画面に表示しているときのスクリーンショットである． OS 管理下のリソースである端末画面に機密情報が出力される場合は，セキュリティマネージャが動的に参照ポインタを真の情報に

自動変換する．これによって，ユーザは端末情報を正しく閲覧可能であることが確認できる．

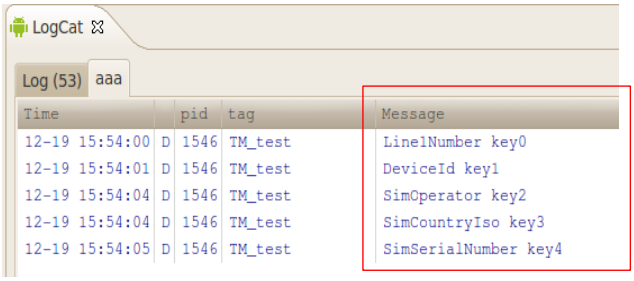

図 5. アプリ側で管理されている実際の値

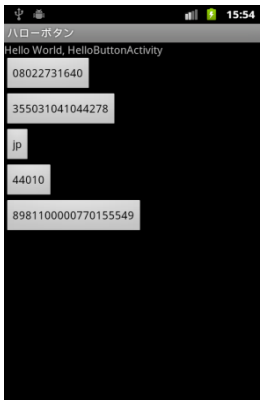

図 6. 機密情報表示画面

5.2 機密情報の SD カードへの出力

端末の電話番号，デバイス ID，SIM の国名，SIM の国コード，SIM 番号を取得し，これを端末に挿入されている SD カードに書き出すアプリを作成し，提案方式を実装した Android フォンにて実行した．

OS の管理外となる SD カードに機密情報を保存するときには，ユーザに対してその可否が尋ねられる（図 7 左）．ユーザの承諾が得られた場合には，セキュリティマネージャが参照ポインタを機密情報に復元した上で，それを SD カードに書き込む（図 7 右上）．もし，承諾が得られなかった場合は，参照ポインタがそのまま SD カードに書き込まれる（図 7 右下）．

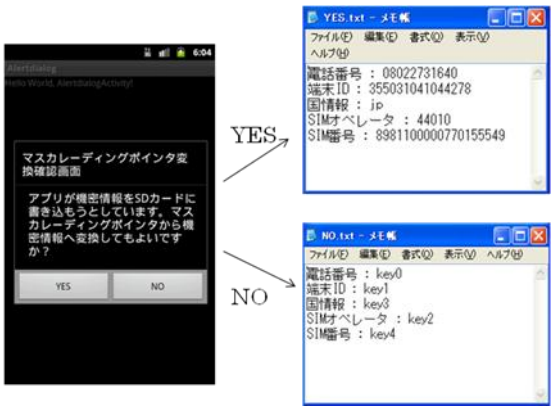

図 7. ユーザ承諾画面と SD カードに保存された機密情報

6. 考察

本方式の適用範囲について考察を行う．

6.1 情報漏洩

提案方式によって，アプリは機密情報そのものを取得することができなくなるため，ユーザの許可無く，自端末の OS 管理外のリソースに機密情報を出力することは不可能になる．本稿では SD カードを介しての情報漏洩を想定し実装を行ったが，その他にもメール（SMS を含む）送信やサーバ接続による情報漏洩に対しても対処可能である．

6.2 不正課金

不正課金については，有料 SMS（SMS メッセージを送信すると課金され，受信者に料金が支払われるサービス）を勝手に送信して課金させるタイプと，web ページにアクセスしただけで，あるいは，web ページ中の画像やリンクなどをクリックしただけで料金を請求するワンクリック型の詐欺がある．それぞれについて考察する．

6.2.1 有料 SMS を勝手に送信するタイプ

端末内の電話帳に登録されている電話番号に有料 SMS を送信するタイプの不正アプリの場合は，アプリが電話帳に登録されている電話番号そのものを入手できないため，提案方式が機能する．しかし，通話先の電話番号をはじめからアプリ内に格納している不正アプリや，通話先の電話番号をその都度，外部から受信するタイプの不正アプリの場合は，OS 内の電話帳にアクセスすることなく有料 SMS を発信するため，提案方式ではこれを防ぐことができない．

6.2.2 ワンクリック詐欺タイプ

ワンクリック詐欺においては，端末情報が記載された架空請求書が不正アプリによって画面に表示される．その際，セキュリティマネージャが行う参照ポインタから機密情報への自動復元処理を一時的にオフにしてやることによって，その請求書の不正をユーザが確認することができる．すなわち，自動復元処理をオフにした際に，画面に表示されている請求書に記載されている端末情報が参照ポインタに変わったとしたら，その請求書を発行した不正者はユーザの端末情報を本当に知っているわけではないということが分かる．ただし，不正者が予めユーザを騙して個人情報を入手し，その情報を記載した偽の請求書を作成した上で，不正アプリを使ってユーザの端末にその請求書を表示させた場合には，提案方式ではこれを防ぐことができない．

6.3 踏み台（bot 化）

端末を操って，端末内の登録されている電話番号，メールアドレス，URL などに勝手にアクセスするタイプの不正

アプリの場合は，アプリが端末内の電話番号，メールアドレス，URL そのものを入手できないため，提案方式が機能する．しかし，アクセス先がはじめからアプリ内に格納されている不正アプリや，アクセス先の情報をその都度，外部から受信するタイプの不正アプリの場合は，OS 内のアドレス情報を取得することなく通信を行うため，提案方式ではこれを防ぐことができない．

6.4 root 権限の不正利用

提案方式のキーコンポーネントであるセキュリティマネージャは OS 管理下にあるため，不正アプリに OS の root 権限が不正に取得されてしまうと提案方式は機能しない．

6.5 ネイティブコード

Android OS では，NDK というツールを使うことで，ネイティブコード（C 言語のソース）をコンパイルし，実行形式ファイル（.so ファイル）を作成することができる．Android アプリは通常 OS の標準 API をコールすることで様々な機能を得るが，.so ファイルをロードすることで Android の標準 API を介さずに直接 Linux 層の機能を使うことができる．

しかし，不正アプリが.so ファイルによって直接 Linux 層の機能を利用した場合も，その不正アプリが OS の root 権限を有していない限り，Linux ファイルシステムのセキュリティ機構によって，OS 内の情報やセキュリティマネージャのメソッドは不正アプリから守られる．

6.6 課題

アプリ側で機密情報のデータを加工したり，機密情報の値を演算に利用する必要がある場合，アプリ自身は機密情報そのものを有していないため，このままではアプリ内でそれらの操作を実行することができない．

7. まとめ

本稿では，現在社会的問題になっているスマートフォンの情報漏洩に対する解決策として，機密情報をマスクするマスカレードポインタの機能を有したセキュリティマネージャを提案し実装した．セキュリティマネージャを Android OS に組み込むことで端末情報の漏洩を防ぐことができる．正規アプリ内での機密情報の加工・演算，不正アプリによる情報漏洩以外の問題の解決，root 権限悪用型アプリへの対策などが今後の課題である．

参考文献

[1] Google Play Store: https://play.google.com/store

[2] TrendLabs Security Blog: http://blog.trendmicro.co.jp/archives/4714

[3] Yomiuri Online”Android 端末に「ボット」出現 ”: http://www.yomiuri.co.jp/net/security/goshinjyutsu/20110107-OYT8T00678.htm

[4] ITmedia, Android OS から見たセキュリティ対策ソフトの制約: http://www.itmedia.co.jp/enterprise/articles/1112/26/news015.html

[5] au Market: http://www.au.kddi.com/seihin/ichiran/smartphone/app/index.html

[6] App Store Review Guidelines - App Store Resource Center: http://developer.apple.com/jp/appstore/guidelines.html

[7] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth : “TaintDroid: An Information - Flow Tracking System for Realtime Privacy Monitoring on Smartphones”, Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI’10), Canada, 2010

[8] Erika Chin, Adrienne Porter Felt, Kate Greenwood, David Wagner; “Analyzing inter-application communication in Android”, Proceedings of the 9th international conference on mobile systems, applications, and services, NY USA, 2011

[9] 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”, コンピュータセキュリティシンポジウム 2011, css2011, 2011.10

[10] 松戸隆幸 児玉英一郎 王家宏 高田豊雄:”Android OS 上でのアプリケーション導入時におけるセキュリティ助言システムの提案”, 情報処理学会研究報告，2012-CSEC, 2012.2

[11] Android: http://developer.android.com/guide/basics/what-is-android.html

[12] ITPro, Android を狙う新たなワンクリ詐欺，ウイルスで料金請求:http://itpro.nikkeibp.co.jp/article/NEWS/20120113/378422/

[13] ITPro, スマホを狙うワンクリ詐欺の新手口, シャッター音や振動で脅かす: http://itpro.nikkeibp.co.jp/article/NEWS/20120312/385782/

付録 A セキュリティマネージャクラス

frameworks/base/telephony/java/android/telephony/SecurityManager.java

```
public class SecurityManager{

        static String[][] keyarray = new String[1000][1000];
                                                //機密情報管理テーブル
        static int i=0; //添字

        //名前 secretinsert
        //引数 参照ポインタ:key，機密情報:secret
        //返り値 無し
        //内容 参照ポインタと機密情報をテーブルに格納するメソッド
         public static void secretinsert(String key,String secret){

                keyarray[i][0] = key;
                keyarray[i][1] = secret;
                i++;

                 }

        //名前 secretsearch
        //引数 参照ポインタ:key
        //返り値 テーブル内の機密情報:keyarray[k][1]
        //内容 参照ポインタから機密情報を検索するメソッド
         public static String secretsearch(String key){

                 int k=0;
                 for(k=0;k<=i;k++){
                                if(key.equals(keyarray[k][0])){
                                                return keyarray[k][1];
                                }
                 }
                 return null;
         }
}
```