平成25年度　修士論文

邦文題目

Androidアプリケーションの挙動を可視化することによるセキュリティ対策の提案

英文題目

Proposal of Security Measures by Visualizing the Behavior of Android Applications

情報工学専攻

(学籍番号: 123430029)

戸田　尚希

提出日: 平成26年1月31日

名城大学大学院理工学研究科

内容要旨

Android 端末の普及に伴って，Android を対象にしたマルウェアが数多く確認されるようになった．Android の今後の更なる普及を考えると，マルウェア対策は重要な課題である．

本論文では，Android 端末において，ユーザにおけるマルウェアの発見やアプリケーションインストール時の安全性の判断を補助するシステムを提案する．本提案では，アプリケーションが GPS 等の機能を利用することや，個人情報を外部サーバへ送信することをユーザに対して可視化する．可視化した情報をもとに，ユーザは危険であると判断したアプリケーションをアンインストールする．同時に，アプリケーションの情報をサーバに送信し，蓄積する．サーバに蓄積された情報は，ユーザがアプリケーションインストール時に参考にし，これからインストールするアプリケーションの安全性の判断を補助する．

今回，アプリケーションの挙動を可視化するビジュアル化プログラムを Android アプリケーションとして実装した．ビジュアル化プログラムでは端末機能である GPS とカメラの利用を検知し，ユーザに通知する仕組みを実現させた．そして，提案方式と市販のセキュリティ対策ソフトを個別に使用した場合及び，両者を併用した場合において定性評価を行った．さらに，提案方式と関連研究においても定性評価を行い，提案方式の有効性を確認した．

目 次

第1章　はじめに

スマートフォンの普及に伴い，多くの人がインターネットを利用する機会が増えた．中でも Android が搭載されたスマートフォンが急速な普及をみせている [1]．Android はオープンソース OS であり，メーカーで独自に拡張が可能である．アプリケーションの開発環境は無料で構築でき，誰でも自由に開発することができる．また，作成したアプリケーションをマーケット上に公開する際に事前審査がないため，誰でも気軽に公開できる．このような利点がある反面，マルウェアと呼ばれる不正なアプリケーションの開発・公開も容易にできてしまう点が課題となっている．近年では，Android 端末をターゲットにしたマルウェアが数多く確認されている [2]．Android の今後の更なる普及を考えると，マルウェア対策は重要な課題である．

Android では，セキュリティモデルとしてパーミッション機構が採用されている [3]．パーミッション機構とは，アプリケーションが利用する機能や情報を，インストール時にユーザに表示して承認を求める仕組みである．アプリケーションは，パーミッションが付与されることにより，保護された機能へのアクセスが可能になる．このため，通常の PC に見られるマルウェアの自動感染の可能性は低く，その面では安全性が高いと言える．

Android におけるマルウェアは，アプリケーションに内包された形で存在する [4]．Android 端末は，マルウェアが内包されたアプリケーションをインストールすることによりマルウェアに感染する．マルウェアは，マルウェアとしての動作を行うために必ずユーザに対してパーミッションを要求する必要がある．そのため，アプリケーションインストール時にユーザがパーミッションを確認することによりマルウェアの感染を防ぐことができる．即ち，インストールするアプリケーションにさえ気を付けていれば，感染の恐れは軽減できる．しかし，ユーザが常にパーミッションを確実にチェックし，正しい判断を行うとは限らない．そのため，パーミッションの表示以外にユーザの判断を補助する手段が必要である．さらに，感染してしまった後についてもそのことを検出し，対策をとる手段も必要である．

Android のマルウェアに関する研究事例は，感染を未然に防ぐ方式 [5–8] と，感染後の被害を抑える方式 [9–14] に大別できる．研究事例は，アプリケーションを提供するマーケット，端末内の Application 層，Application Framework 層，Linux Kernel 層，のどこで主な対策を行っているのかという点について分類することができる．

マーケットでの対策例として，文献 [5] がある．文献 [5] では，アプリケーションを提供するマーケットが安全になれば，多くのユーザの Android 端末を保護できるという考えの下，Android アプリケーションからなる事前審査ツールを提案している．この提案は，マー

ケットの規模によって，アプリケーションの評価を行う評価者に負担がかかるという課題がある．

Application 層での対策例として，文献 [6, 7] がある．文献 [6] では，法人向けの Android 端末のセキュリティ保護を目的として，脅威の低いアプリケーションのホワイトリストを作成し，インストールするアプリケーションが安全かどうかを判断する仕組みを提案している．この提案では，インストールするアプリケーションが限られる法人向けの Android 端末を対象に提案方式が考えられている．個人向けの Android 端末では，多くのアプリケーションのインストールが予想されるため，ホワイトリストに定義されるアプリケーション数が増加することが考えられる．その結果，ホワイトリストの管理が困難となるため，個人向けのセキュリティ対策とするには適さない．文献 [7] では，アプリケーションインストール時にパーミッションに基づいた危険性検知と危険性提示を行うことにより，ユーザのアプリケーション導入の判断支援を行う仕組みを提案している．この提案では，危険性検知の誤検知率の高さが課題である．

Application Framework 層での対策例として，文献 [8–11] がある．文献 [8] では，アプリケーションによって要求されるパーミッションを条件付で許可する等，ユーザが許可するパーミッションを選択できるフレームワークを提案している．この提案では，パーミッションを拒否してもアプリケーションを実行できるため，アプリケーションによっては，正常に動作しない可能性があるという課題がある．文献 [9] では，個人情報等を扱う API への割り込みと，割り込みを受ける制御アプリケーションを Android に実装する，API ごとのリアルタイム動的制御方式を提案している．この提案では，Application Framework を改造する必要があるためシステム導入の敷居が高い．文献 [10] では，アプリケーションに機密情報を保持させない仕組みを提案をしている．この提案では，アプリケーション自体が機密情報を持っていないため，機密情報の加工をすることができないという課題がある．文献 [11] では，WebView の脆弱性を利用した攻撃を防止するアクセス制御方式を提案している．この提案も，Application Framework を改造する必要があるためシステム導入の敷居が高い．

Linux Kernel 層での対策例として，文献 [12–14] がある．文献 [12] では，アプリケーションにより送信される HTTP パケットを Linux Kernel が独自の情報フロー制御アプリケーションに転送し，そのパケットを解析することにより，個人情報や位置情報等の機密情報が含まれていた場合に通信制御を行う提案をしている．この提案では，パケット内の情報を解析する必要があるため，暗号化されたパケットへは提案方式を適用できない．文献 [13] では，個人情報の追跡が出来るテイント解析を利用した解析システムを提案している．この研究を発展させた研究として文献 [14] がある．これらの提案は，Linux Kernel を改造する必要があるためシステム導入の敷居が高い．

本研究では，Android セキュリティについての調査や上記のような関連研究の調査を行った結果，システム導入が容易で，アプリケーションのインストール前後でマルウェア対策を行う提案方式を考えるに至った．

本論文では，ユーザによるマルウェアの発見やインストール時にアプリケーションの安全性の判断を補助するシステムを提案する．マルウェアが及ぼす被害の 1 つである情報漏洩被害は，端末機能による情報の取得と外部送信により成り立つ．そのため，情報の「取得」と「送信」のどちらかでその動作の正当性をユーザに確認すれば，ユーザによる情報漏洩の防止が実現できる．提案方式は，アプリケーションによる GPS やカメラ機能の利用や個人情報の外部サーバへの送信を可視化することにより，ユーザによるマルウェアの発見を可能とする．可視化された情報をもとに，ユーザはアプリケーションの正当性を判断し，必要であればアンインストールする．また，ユーザが危険であると判断したアプリケーションは，要注意アプリケーションとしてその情報をサーバに送信して蓄積する．サーバに蓄積された情報は，アプリケーションインストール時に確認することができ，これからインストールするアプリケーションの安全性の判断を行うユーザを補助する．

今回，アプリケーションの挙動を可視化するビジュアル化プログラムを Android アプリケーションとして実装した．ビジュアル化プログラムでは端末機能である GPS とカメラの利用を検知し，ユーザに通知する仕組みを実現させた．実装を行うにあたり，パーミッションの組み合わせによって起こる被害と実在するマルウェアが及ぼす被害を参考にして，提案方式で対象とする被害を定義した．今回はその中でも，位置情報と写真データの情報漏洩に着目して実装を行った．そして，提案方式と市販のセキュリティ対策ソフトを個別に使用した場合及び，両者を併用した場合において定性評価を行った．さらに，提案方式と関連研究においても定性評価を行い，提案方式の有効性を確認した．

以下，2 章で Android セキュリティの課題を述べる．3 章で関連研究についての説明を行い，4 章で提案方式について述べる．また，5 章で実装について説明し，6 章で評価について述べる．7 章でまとめる．

第2章　Androidセキュリティの課題

2.1　パーミッション機構の課題

Android にはセキュリティ面を考慮して，パーミッション機構という仕組みがある．パーミッション機構とは，アプリケーションインストール時にそのアプリケーションが要求しているパーミッション一覧をユーザに表示した上で，インストールの許可を求める仕組みである．これは，マルウェア対策に特化した仕組みではないため，マルウェアに関してはセキュリティ対策が不十分である．パーミッション機構の課題としては以下の点が挙げられる．

課題 1 インストール時に表示されるパーミッションから，マルウェアを発見するのは難しい．

課題 2 インストールしたマルウェアを見つけ，被害の拡大を防ぐことは難しい．

Android において，マルウェアの感染を防ぎ，被害を食い止めることができる可能性があるのは，アプリケーションのインストール時である．マルウェアを感染前に防ぐことができれば，Android におけるセキュリティ対策の中で最も効力のある対策である．しかし，現状のパーミッション機構では，アプリケーションのインストール時に表示されるパーミッションから，ユーザがアプリケーションの動作を予想してマルウェアであるかを判断するのは困難である．また，アプリケーションインストール後については，マルウェアを検知して対策を施すことは行われない．そのため，マルウェアが一旦インストールされてしまえば，GPS 等の機能の利用や個人情報の送信等が行われていてもユーザは知ることができない．Android にてセキュリティ対策を施すには，Android 端末が持つ快適性・利便性を損なわずに，マルウェア感染前後でそれぞれ対策を考える必要がある．

2.2　市販のセキュリティ対策ソフトの課題

市販のセキュリティ対策ソフトでは，以下のようなマルウェア対策が行われている．本節では市販のセキュリティ対策ソフトの例として，ノートンモバイルセキュリティを挙げる．ノートンモバイルセキュリティのマルウェア対策の主な機能としては，以下の 4 点が挙げられる [15].

- **ダウンロードの脅威からの保護**
  アプリケーションの更新やインストールをする際，自動的にスキャンすることにより，パフォーマンスに影響を与えることなく脅威からの感染を未然に防ぐ．

- **着信と SMS の遮断**
  「特定の電話番号からの着信」,「迷惑な SMS」,「危険性の高い見知らぬ人からのメッセージ」を遮断する.

- **SD メモリカードのスキャン**
  デバイスに差し込む SD メモリカードを自動的にスキャンすることにより，脅威からの感染を未然に防ぐ.

- **ノートンコミュニティウォッチ**
  世界中のノートンユーザーから寄せられた情報を，新たな脅威の迅速な発見や安全なファイルの識別，保護機能の強化や効率化に役立てる.

上記 4 点の中でも特徴的な機能は，ノートン独自の機能であるノートンコミュニティウォッチである．ノートンコミュニティウォッチを通して収集された情報は，各ユーザの端末で実行されたファイルのセキュリティ評価レーティングを計算するアルゴリズムによって評価される．その結果を，新たなパターンやヒューリスティックとしてセキュリティ対策ソフトに提供することにより，未知のマルウェアに対応しようとしている.

セキュリティ対策ソフトには様々なマルウェア対策の機能がある一方で，課題になっていることがある．それが以下の 2 点である.

- **新種や亜種のマルウェアの検出が不十分であること**
  パターンマッチングがマルウェア検出の主流であるため，定義ファイルに定義されていないマルウェアは検出できない．また，ヒューリスティック検知においても検出できないマルウェアが存在する.

- **保存先によってはセキュリティ対策ソフトのスキャンができないこと**
  セキュリティ対策ソフトは，ほとんどが一般権限で動作する．しかし，一般権限では「/data/app」もしくは「SD カード」しかスキャンできない[1]．そのため,「/data/app_private」や「/system」へインストールされたマルウェアに関しては，セキュリティ対策ソフトでは検出・削除ができない [16].

文献 [16] によると，/data/app_private にアプリケーションがインストールされるのは，アプリケーション開発者がマーケットに公開する際に「コピー防止オプション」を有効にした場合である．これが有効になっているとアプリケーションの APK ファイルがユーザに読み取り権限のない/data/app_private に格納される．通常のアプリケーションと同じ一般権限のセキュリティ対策ソフトは/data/app_private 内のファイルを読み取れないため[2]，スキャンができない．このことから,「コピー防止オプション」を有効にしたマル

[1] Android では，/data/app，/data/app_private，SD カードのどこかにアプリケーションがインストールされる．通常は/data/app にインストールされるようになっている.

[2] root 化した Android 端末ならばファイルの読み取りは可能.

ウェアがマーケット上に公開されていた場合，セキュリティ対策ソフトのスキャンを潜り抜けることができる．

2.3 マルウェア被害の分類

表 2.1 に既存のマルウェアが Android 端末に対してどのような被害を及ぼしているかを分類した．表 2.1 は，文献 [17] をもとに独自の判断で分類したものである．マルウェアが及ぼす被害は，情報送信被害，端末内における被害，ダウンロード被害，その他の被害に分類できる．情報送信被害は，端末内の情報を外部へ送信する被害である．端末内における被害は，端末内部で起こる被害である．ダウンロード被害はマルウェアが有害なアプリケーションを勝手にダウンロードしようとする被害である．この被害は，端末内における被害と情報送信被害の発生に繋がる可能性を持っている．その他の被害とは，上記分類に当てはまらない被害である．

上記の分類の中では，情報送信被害と端末内における被害の原因となるマルウェアが数多く報告されている [17] ．この 2 種類の被害を比較すると，ユーザにとって最も深刻な被害は情報送信被害である．Android 端末は PC よりも端末の利用頻度が高い分，位置情報やアドレス帳等，よりプライベートな情報が保存されている．このような情報が外部へ漏れることにより犯罪などに利用される可能性もあることから，情報送信被害をユーザにとって最も深刻な被害とした．情報送信被害を発見し，被害を防ぐことが Android 端末における最大の課題を解決することに繋がる．

表 2.1 マルウェアによる被害の分類

被害の種類	構成要素
情報送信被害	IMEI，IMSI をサーバに送信
	SMS メッセージを外部サーバに送信
	プレミアム SMS の番号への SMS 送信
	位置情報をサーバに送信
	電話番号，連絡先リストをサーバに送信
	写真撮影と撮影した写真をサーバに送信
端末内における被害	ホーム画面にショートカットを追加
	ネットワーク設定の変更
	他の実行中のプロセスの強制終了
	特定の SMS メッセージの削除
	通話内容を記録して SD カードに保存
	端末の再起動
ダウンロード被害	アプリケーションを/system にインストール
	アプリケーションをアンインストール
その他の被害	送られてくる SMS メッセージの監視
	ルート権限の取得

2.4 マルウェアに取得されたくない情報

Android には数多くのパーミッションが存在する．マルウェアが及ぼす被害は，複数のパーミッションを組み合わせたことが原因であることが多い．例えば，情報漏洩被害は，情報を外部へ送信することを許可するパーミッションと，位置情報など端末内の情報を取得するパーミッションの組み合わせによって起こる．

文献 [16] を参考に，情報漏洩を引き起こす可能性のあるパーミッションの組み合わせを表 2.2 にまとめた．表 2.2 における利用されるパーミッションとは，情報を外部へ送信することを許可するパーミッションである INTERNET や SEND_SMS と組み合わせることを前提としている．そして，実際に存在するマルウェアが及ぼす被害をまとめた表 2.1 と表 2.2 を照らし合わせた．その結果，表 2.2 における太字下線部の被害内容と悪用されやすいパーミッションの組み合わせが浮かび上がった．本研究では，これらのパーミッションに保護された情報である，IMEI，IMSI，SMS メッセージ，位置情報，連絡先リスト，電話番号，写真データをマルウェアに取得されたくない情報であると結論付けた．

表 2.2　パーミッションの組み合わせによって想定される情報漏洩被害

想定される情報漏洩被害	利用されるパーミッション
アカウント ID（Google の場合は Gmail アドレス）が外部へ送信される	GET_ACCOUNTS
実行時に確認メッセージで許可を求められるが，サービス内のデータが取得され，外部へ送信される	USE_CREDENTIALS
受信ボックス内の SMS が読み出され，外部へ送信される	**READ_SMS**
受信時に SMS の内容が取得され，外部へ送信される	**RECEIVE_SMS**
受信時に MMS の内容が取得され，外部へ送信される	RECEIVE_MMS
受信時に WAP-PUSH の内容が取得され，外部へ送信される	RECEIVE_WAP_PUSH
住所録の情報が読み出され，外部へ送信される	**READ_CONTACTS**
予定表の情報が読み出され，外部へ送信される	READ_CALENDAR
ユーザ辞書の内容が読み出され，外部へ送信される	READ_USER_DICTIONARY
ユーザプロファイル情報が読み出され，外部へ送信される	READ_PROFILE
システムログが読み出され，外部へ送信される	READ_LOGS
電話番号，SIM 番号，着信電話番号などの情報が読み出され，外部へ送信される	**READ_PHONE_STATE**
入力したパスワードなどが読み取られ，外部へ送信される	READ_INPUT_METHOD
盗聴したデータを外部へ送信	RECORD_AUDIO
盗撮したデータを外部へ送信	**CAMERA**
ユーザの現在位置を収集し，外部へ送信	**ACCESS_FINE_LOCATION**, **ACCESS_COARSE_LOCATION**
通話を監視し，電話をかけた時間や電話番号などを外部へ送信	PROCESS_OUTGOING_CALLS
MAC アドレスを取得し，外部へ送信	ACCESS_WIFI_STATE
ブラウザのブックマークを収集し，外部へ送信	READ_HISTORY_BOOKMARKS
アプリケーションの使用状況を収集し，外部へ送信	GET_TASKS
バッテリーの使用状況を収集し，外部へ送信	BATTERY_STATS

2.5 ユーザのセキュリティ意識

ユーザは，Android 端末を従来の携帯電話を扱っている通信キャリアから購入する．そのため，ユーザは従来の携帯電話と同じような感覚で Android 端末を利用しているのが現状である．ユーザのセキュリティ意識の低さが表れている例として，端末のパスワード設定をしない，セキュリティ対策ソフトを使用しない，ファイルのバックアップをしない等，予防策を行っていないことが挙げられる [18]．このように，Android 端末を利用するユーザのセキュリティ意識や危機感の低さもマルウェア被害を拡大する原因として問題となっている．

第3章　関連研究

3.1　関連研究の分類

図 3.1 に Android のセキュリティに関連する研究の一覧を示す．Android は，Application，Application Framework，ライブラリ，Android Runtime，Linux Kernel という 5 つのスタックから構成されている [19]．今回調査した関連研究は，端末内の Application，Application Framework，Linux Kernel でセキュリティ対策が行われていた．他にも，端末外のマーケットでセキュリティ対策が行われている研究もあった．

マーケットとは，アプリケーションの提供を行う役割がある．代表的なマーケットとしては，Google play が挙げられる．Application とは，電話やカレンダーなどユーザが実際に操作するアプリケーションを扱う役割がある．Application Framework とは，アプリケーションのライフサイクルの管理やユーザインタフェースの表示などをアプリケーションに提供する役割がある．Linux Kernel とは，メモリやプロセス管理，ファイルシステムなどの機能を提供する役割がある．図 3.1 は，関連研究で提案されている技術が，アプリケーションを提供するマーケット，端末内の Application 層，Application Framework 層，Linux Kernel 層のどこで主な対策を行っているのかという点についてまとめたものである．

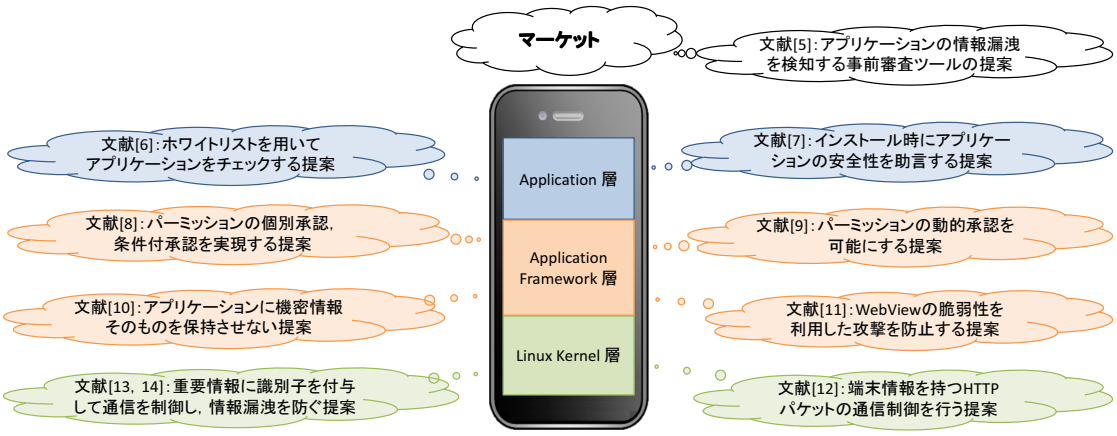

図 3.1　関連研究のまとめ

3.2　マーケットにおける対策

文献 [5] では，アプリケーションを提供するマーケットが安全になれば，多くのユーザの Android 端末を保護できるという考えの下，Android アプリケーションからなる事前審査ツールを提案している．図 3.2[1] にこの提案における情報漏洩検知までの流れを示す．

この提案では，情報漏洩の検知をするにあたり，Logcat ログに注目している．評価者は，アプリケーションのインストール後，アプリケーションを起動して Logcat ログの取得を開始する．「ActivityManager Start proc」から始まる Logcat ログからアプリケーションの起動を検知し，プロセス ID を取得する．取得したプロセス ID と一致する Logcat ログと正規表現で表現された重要情報の組み合わせシグネチャ，広告シグネチャとを照らし合わせて，情報漏洩の痕跡や広告表示の痕跡を検知する．検知結果によって，アプリケーションをマーケットに公開するか否かを決める．この提案は，マーケットの規模によって，アプリケーションの評価を行う評価者に負担がかかるという課題がある．さらに，マーケットにアプリケーションの事前審査の仕組みを適用することは難しいという課題もある．

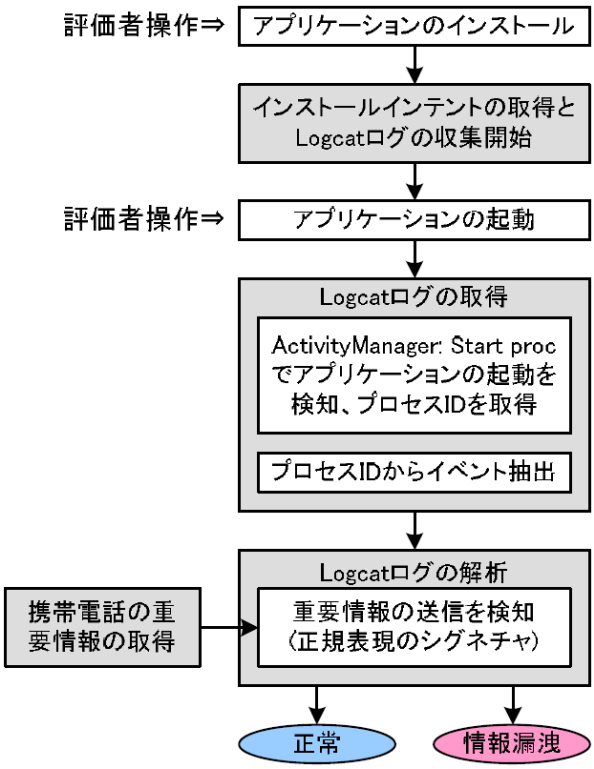

図 3.2　情報漏洩を検知するまでの手順

[1]図 3.2 は文献 [5] から引用.

3.3 Application 層における対策

文献 [6] では，法人向けの Android 端末のセキュリティ保護を目的とした研究が行われている．図 3.3[2] にこの提案の概要を示す．この提案では，法人管理者が予め業務アプリケーションを収集して実行し，その動作ログを安全性の評価を行うセンター局へ送信する．センター局は動作ログの解析を行い，安全であると認定したアプリケーションのホワイトリストを作成して，業務用端末に配信する．各端末では，独自のアプリケーションがこのホワイトリストをチェックし，インストールするアプリケーションが安全かどうかを判断する．アプリケーションが認定されていない場合，社員に対してアンインストールを促す．この提案は，インストールするアプリケーションが限られる法人向けの Android 端末を対象としている．個人向けの Android 端末では，多くのアプリケーションのインストールが予想されるため，ホワイトリストに定義されるアプリケーション数が増加することが考えられる．その結果，ホワイトリストの管理が困難となるため，個人向けの Android 端末のセキュリティ対策とするには適さない．

文献 [7] では，パーミッションに基づいた危険性検知と危険性提示を行うことにより，アプリケーション導入のユーザの判断支援を行う．この提案では，パーミッションの組み合わせに着目し，パーミッションの組み合わせによって起こる位置情報の流出等の危険性をユーザへ提示する．図 3.4[3] にユーザへ提示するインタフェースのイメージを示す．これにより，アプリケーションをインストールする際のユーザの判断を補助する．この提案では危険性検知において，マルウェア 180 個，Android マーケット上から入手したアプリケーション 261 個を用いて評価を行ったところ，マルウェアの検知率が 95%と高い検知率であった．しかし，正規のマーケットから入手したアプリケーションも 62 %を危険性のあるアプリケーションとして検知した．このように，この提案では誤検知率の高さが課題となっている．

3.4 Application Framework 層における対策

文献 [8] では，アプリケーションが要求する全てのパーミッションを一括許可しなければアプリケーションをインストールできないという問題点を解決している．この提案では，アプリケーションによって要求されるパーミッションを条件付で許可する等，ユーザが許可するパーミッションを選択できるフレームワークを提案している．図 3.5[4] に許可するパーミッションを選択できる画面を示す．この提案を適用すれば，1 つ 1 つのパーミッションに対して「常に許可する」,「常に拒否する」,「条件付きで許可する」の 3 つから選択することができる．ただし，パーミッションを拒否してもアプリケーションを実行できるため，アプリケーションによっては，正常に動作しない可能性があるという課題がある．

[2]図 3.3 は文献 [6] から引用.
[3]図 3.4 は文献 [7] から引用.
[4]図 3.5 は文献 [8] から引用.

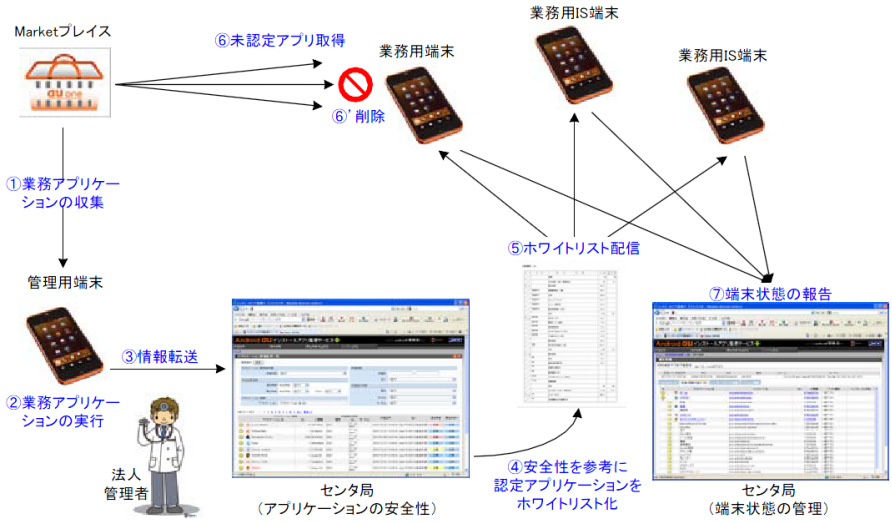

図 3.3　ホワイトリストを用いた法人向けのセキュリティ対策の概要

図 3.4　ユーザに表示するインタフェースのイメージ

文献 [9] では，個人情報等を扱う API への割り込みと，割り込みを受ける制御アプリケーションを Android に実装することによって，API ごとのリアルタイム動的制御方式を提案している．この提案では，パーミッションに保護されている個人情報を扱う API へフックポイントを設ける．アプリケーションが実行されてフックされると， 拡張された Application Framework により，独自の制御アプリケーションによって定義されたセキュリティポリシーリポジトリが参照される．その後，ユーザに対して個人情報や位置情報へのアクセスを許可するか否かを求めるアラームが表示される．図 3.6[5] の中央がパーミッションの動的承認を行う際に表示されるアラームである．左図が，アプリケーション「Maps」へパーミッションを与えることを許可した場合の図である．現在位置が示されていることが確認できる．ま

[5]図 3.6 は文献 [9] から引用.

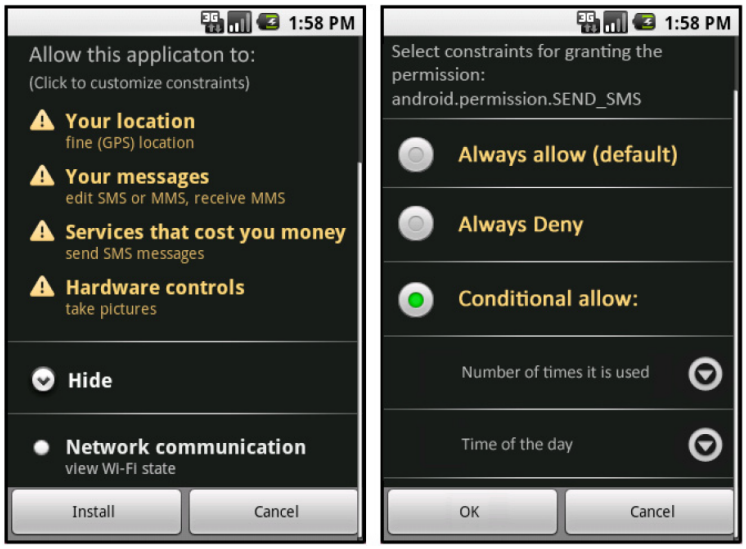

図 3.5　パーミッションの個別承認や条件付き承認を行う画面

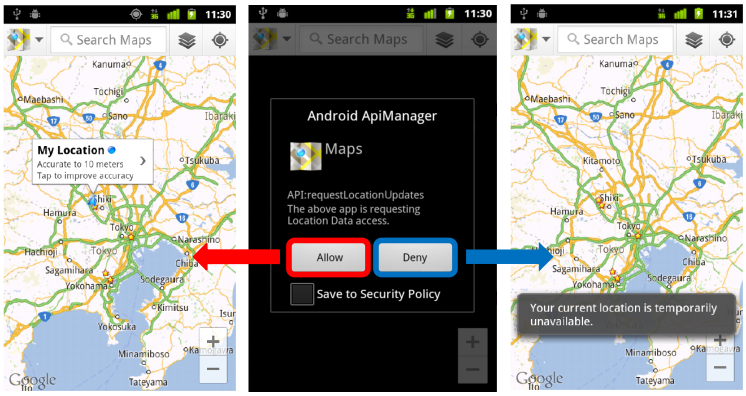

図 3.6　パーミッションの動的承認を行うアラーム

た，右図がアプリケーション「Maps」へパーミッションを与えることを拒否した場合の図である．現在位置情報が利用できないことが確認できる．表示されるアラームは，セキュリティポリシーの設定によって非表示にできる．

文献 [10] では，アプリケーションに機密情報を保持させない仕組みを提案をしている．この提案では，Application Framework にセキュリティマネージャという機構を設ける．アプリケーションが機密情報を扱う API を呼び出した時に，セキュリティマネージャは参照ポインタを生成し，機密情報と対応付ける．そして，その参照ポインタを機密情報の代わりにアプリケーションへ渡す．アプリケーションが端末外へ情報の送信を行う際は，ユーザに参照ポインタを機密情報へ変換する可否を問うアラームを提示する．図 3.7[6] にユーザに提示するアラームを示す．ユーザが参照ポインタを機密情報へ変換することを許可した場合，図 3.7 の右上図のように参照ポインタは機密情報に変換されてアプリケーション[7] へ渡される．ユー

[6]図 3.7 は文献 [10] から引用.
[7]機密情報を取得してその内容をファイルへ出力するアプリケーション.

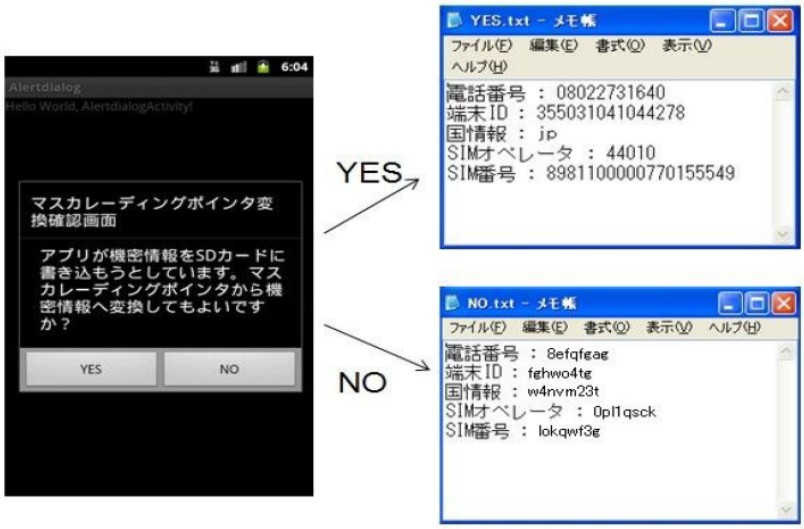

図 3.7　参照ポインタの機密情報への変換

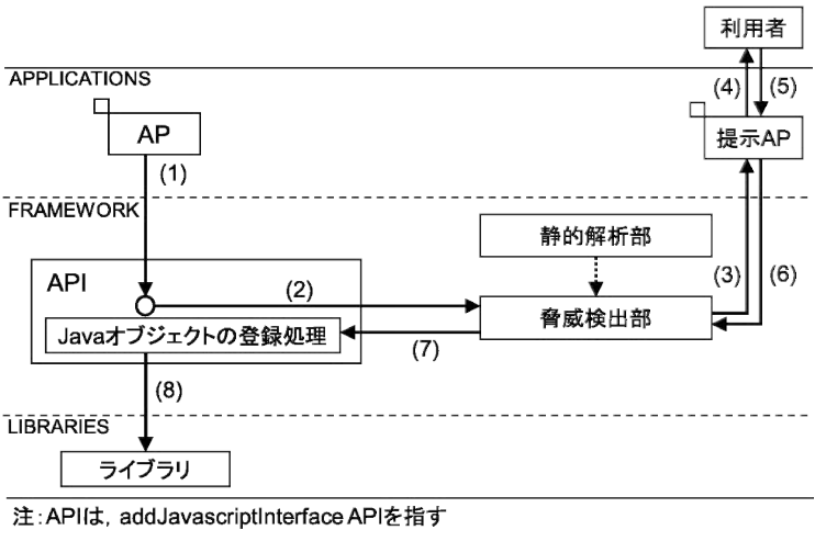

図 3.8　WebView の脆弱性を利用した攻撃を防止する提案方式の概要

ザが参照ポインタを機密情報へ変換することを拒否した場合，図 3.7 の右下図のように参照ポインタそのものがアプリケーションへ渡される．この提案では，アプリケーション自体が機密情報を持っていないため，機密情報の加工をすることができないという課題がある．

文献 [11] では，WebView の脆弱性を利用した攻撃を防止するアクセス制御方式を提案している．この提案では，Android アプリケーションが Javascript と連携する時に利用する Java オブジェクトにアクセス制御機構を追加する．この提案の流れを図 3.8[8] に示す．アプリケーションは，addJavascriptInterface API を呼び出す．そして，Java オブジェクトの登録処理を行う際に，Android アプリケーションを逆アセンブルする．Java オブジェクトが実行できる

[8]図 3.8 は文献 [11] から引用.

処理を解析し，危険な API リストと比較する．これにより，Java オブジェクトの利用により生じる潜在する脅威を検出できる．さらに，脅威検出部が，脅威があると判断した場合は提示 AP を呼び出す．そして，Web ページの JavaScript から Android 端末に定義されたメソッドの実行の可否をユーザに問うアラームを提示する．ユーザは提示された情報を参考に可否判断を行う．提示 AP は判断結果を脅威検出部に返却し，脅威検出部は判断結果を addJavascriptInterface API に返却する．ユーザが脅威を感じた場合，Java オブジェクトを無効化して Java オブジェクトの登録処理を行わず，addJavascriptInterface API の処理を終了する．その結果，Javascript による Android 端末に定義されたメソッドの実行を防ぎ，Android 端末への攻撃を防止している．この提案は，Application Framework を改造を必要とし，システム導入の敷居が高い．

3.5 Linux Kernel 層における対策

文献 [12] では，アプリケーションの動作ではなく，送信されるパケットに注目した提案をしている．図 3.9[9] に情報フロー制御アプリケーションの概要を示す．監視対象アプリケーションはパーミッションで許可された端末上の情報を取得する．監視対象アプリケーションにより送信される HTTP パケットは，Linux Kernel により独自の情報フロー制御アプリケーションに転送され，情報フロー制御アプリケーションはそのパケットを解析する．個人情報や位置情報等の機密情報が含まれていた場合に通信制御を行う．この提案は，Linux Kernel を改造する必要があり，システム導入の敷居が高いという課題がある．また，パケット内の情報を解析する必要があるため，暗号化されたパケットへは適用できない．

文献 [13] では，個人情報の追跡が出来るテイント解析を利用した解析システムを提案している．図 3.10[10] にテイント解析における構造図を示す．テイント解析は，テイントが付与される情報群を示す source，テイントの伝搬を示す propagation，伝搬されたテイントを監視するポイントを示す sink の 3 つで構成されている．テイント解析とは，source と呼ばれる情報にテイントという識別子を付与し，その情報が使用された場合は使用された情報と共に伝搬するテイントの流れを解析する手法である．この提案では，電話帳などデータベースに格納された個人情報へのテイントの付与が不十分であり，これらのデータの送信を検知できないという課題があった．この課題を解決したのが文献 [14] である．文献 [14] では，従来手法では存在しなかったデータベース情報の送信検知を実現し，テイント付与の範囲を拡大することにより，この課題を解決している．しかし，文献 [13] も文献 [14] も Linux Kernel を改造する必要があり，システム導入の敷居が高い．

[9]図 3.9 は文献 [12] から引用.
[10]図 3.10 は文献 [14] から引用.

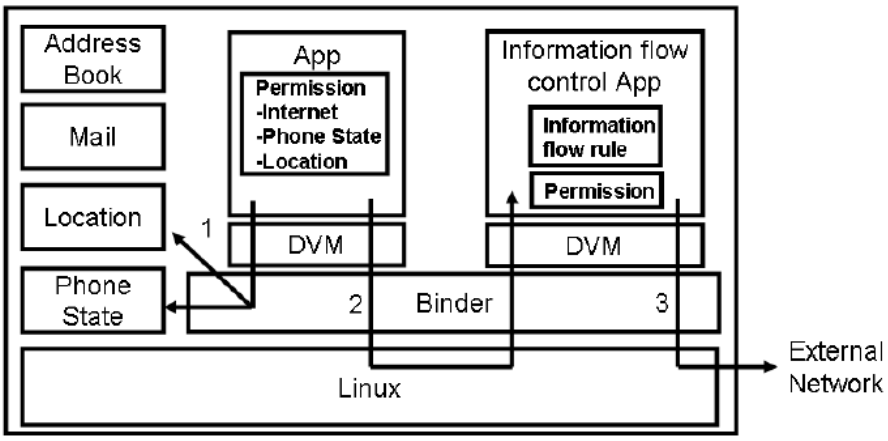

図 3.9　情報フロー制御アプリケーションの概要

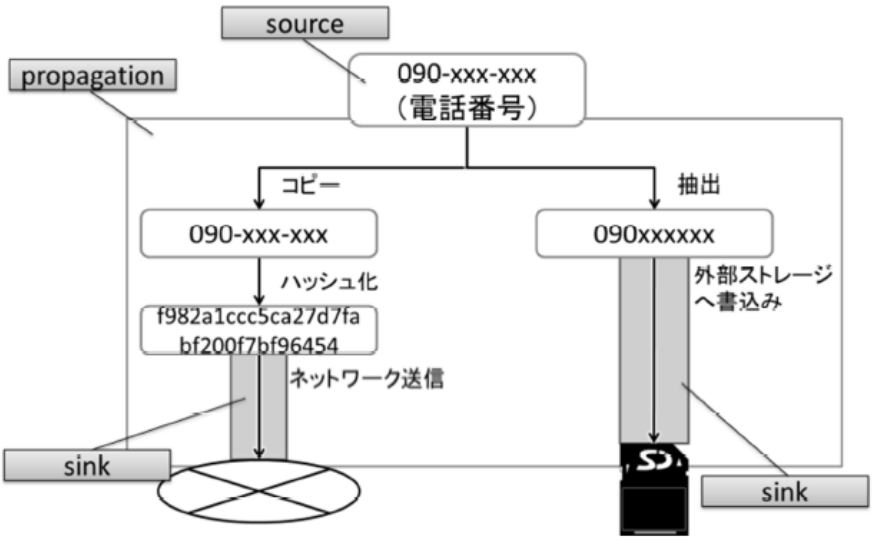

図 3.10　テイント解析における構造図

第4章　提案方式

4.1　提案にいたる背景

本提案では，アプリケーション層で出来る限りのマルウェア対策を行うことを想定している．本提案におけるセキュリティ対策の精度を向上させるには，より多くの人に提案方式が適用された端末を利用してもらう必要がある．そこで，本提案では Android のバージョンや機種に関わらず，容易に適用が可能であるアプリケーション層での対策を試みた．そして，本提案を市販のセキュリティ対策ソフトと併用することにより，お互いの弱点を補強し合うことによって Android セキュリティの向上を図る．

Android では，アプリケーションのインストール時にパーミッションを要求する．マルウェアも正規のアプリケーションと同様にパーミッションの要求を行い，ユーザに自分自身をインストールさせて，情報の漏洩等の被害を引き起こす．ユーザは，アプリケーションから要求されるパーミッションの必要性を正しくチェックできれば，マルウェアをインストールすることはない．しかし，広告表示のためなどの正当な理由で，本来の目的以上のパーミッションを要求する正規アプリケーションも存在する．そのため，Android を対象にしたマルウェアをシステム側で対策を施すには限界があると考え，本提案ではユーザの補助を必要とする方式とした．

本提案で対象とする被害は表 4.1 の通りで，これは表 2.2 や 2.4 節のマルウェアに取得されたくない情報をもとにしている．情報漏洩は，端末機能による情報の取得と外部送信により成り立つ．そのため，情報の「取得」と「送信」のどちらかでその動作の正当性をユーザに確認すれば，ユーザによる情報漏洩の防止が実現できる．表 4.1 の被害の中でも，Logcat ログによる端末機能の検知が可能であった位置情報の流出と写真データの流出を防ぐことに着目した．

表 4.1　本提案で対象とする被害内容

パーミッション 1	パーミッション 2	被害内容
INTERNET	READ_CONTACT	連絡先リストの流出
INTERNET	CAMERA	写真データの流失
INTERNET	READ_PHONE_STATE	電話番号や IMEI 等の流出
INTERNET	ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION	位置情報の流出
SEND_SMS	READ_SMS，RECIEVE_SMS	意図しない SMS の送信や受信

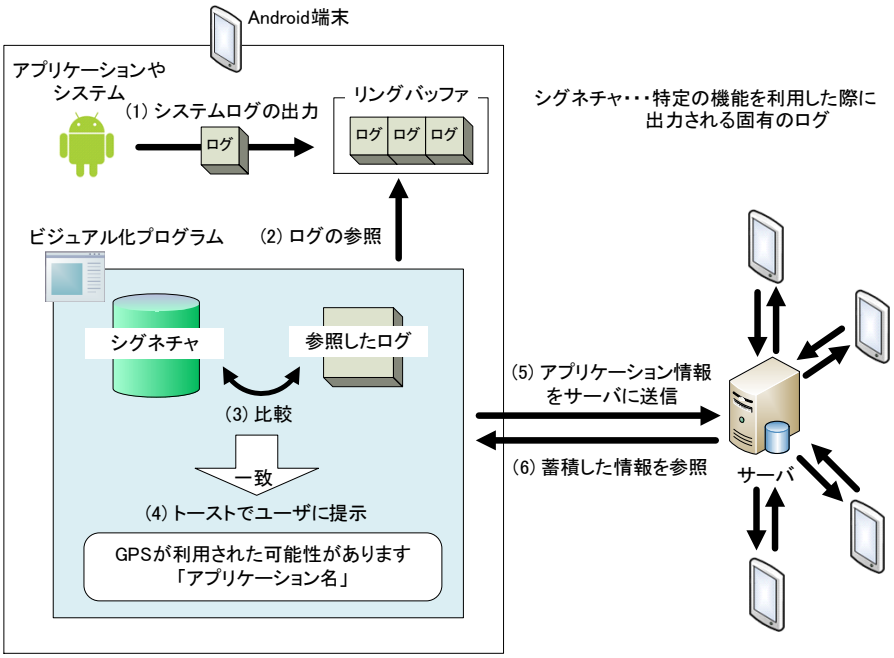

図 4.1　提案方式の動作概要

4.2　提案方式の動作概要

本提案では，Android 上で動作するアプリケーションによる GPS 等の機能の利用や個人情報の外部サーバへの送信をユーザに対して通知する．その結果，ユーザが判断した要注意アプリケーションの情報をサーバに送信し，蓄積する．この情報をもとに，複数ユーザのアプリケーションインストール時における安全性の判断を補助する．

図 4.1 に提案方式の動作概要を示す．本提案では，アプリケーションが GPS 等の機能を利用する際に出力される固有のログをシグネチャとして予め定義する．(1) Android では，アプリケーションやシステムがログを出力し，リングバッファに一時的に保存する．(2) アプリケーションの挙動を可視化するために，Android アプリケーションとして実装されたビジュアル化プログラムが，Logcat コマンドを用いてこのバッファに保存されたログを参照する．(3) この際に，参照したログとシグネチャを比較する．(4) 一致した場合，GPS 等の機能が利用された，または個人情報が送信されたとみなし，トーストを表示する．トーストとは，画面に一時的に表示するユーザ通知機能である．トーストには，アプリケーションの挙動の内容とそのアプリケーション名を表示する．ここでは，端末購入時にインストール済みのアプリケーションに関しては安全であることを前提としている．そのため，プリインストールされているアプリケーションの挙動に関してはユーザへの提示は行わない．ユーザはその表示が意図していない内容であった場合，アプリケーションを削除すべきかどうかを判断する．(5) 同時に，ユーザが要注意であると判断したアプリケーション情報をサーバに送信する．提案方式では悪意のある人が，真に注意が必要な情報を隠ぺいするために，多くの偽情報をサーバに送信することが考えられる．この対策として，1 つのアプリケーションに対す

るサーバへの送信を 1 人 1 件に限定することにより，偽情報の大量アップロードを防止できると考えている．

サーバには，要注意アプリケーション情報を蓄積して，複数のユーザで情報を共有する．(6) 新たにアプリケーションをインストールする際に，このサーバに蓄積された情報を参照して参考にすることにより，これからインストールするアプリケーションが他のユーザからどのように思われているのかを知ることができる．その結果，アプリケーションのインストールを控えるなど，ユーザ自身でマルウェアへの対策をとることができる．

4.3 Logcat ログの調査

本提案では，Android 端末にインストールされているアプリケーションの挙動をアプリケーションレベルで検知する方法として Logcat ログに着目した．マルウェアに取得されたくない情報として 2.4 節で定義した項目の中から，Logcat ログにより挙動の検知が可能である項目について調査した．Logcat ログの確認をした実機は，Android2.2 を搭載したシャープ (株) の IS03 と Android 4.0 を搭載した HTC Corporation の ISW13HT を使用した．Logcat ログには，アプリケーション自体から出力するログと，システムから出力されるログの 2 種類がある．前者は，内容がアプリケーションに依存するため，後者に注目して出力ログの解析を行った．その結果，Logcat ログにより GPS の利用とカメラの利用が検知できることを確認し，提案方式におけるシグネチャを決定した．

4.3.1 GPS の利用を検知する Logcat ログ

図 4.2 は Android2.2 を搭載したシャープ (株) の IS03 において，位置情報を取得するアプリケーションを実行した際に確認できたログの一部である．図 4.2 の下線部の「GpsLocationProvider」とは，GPS を利用する際に使用されるクラスである．Android4.0 を搭載した HTC Corporation の ISW13HT においても同様のログを得られたことから，GPS の利用を検知するには「GpsLocationProvider」に関するログをシグネチャとして利用できると考えられる．

4.3.2 カメラの利用を検知する Logcat ログ

図 4.3 は Android2.2 を搭載したシャープ (株) の IS03 において，カメラを利用するアプリケーションを実行した際に確認できたログの一部である．図 4.3 の下線部の「ActivityManager」，「act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER」，「android.camera」，「Displayed」というキーワードが Android4.0 を搭載した HTC Corporation の ISW13HT であっても確認できた．そのため，カメラの利用を検知するには上記キーワードに関するログをシグネチャとして利用できると考えられる．

```
2012-09-16 22:30:14.323 D 280/GpsLocationProvider: setMinTime 0
2012-09-16 22:30:14.323 D 280/GpsLocationProvider: startNavigating
2012-09-16 22:30:14.343 D 280/GpsLocationProvider: Acquiring wakelock
2012-09-16 22:30:42.463 D 280/GpsLocationProvider: TTFF: 28122
2012-09-16 22:30:45.993 W 18051/KeyCharacterMap: Can't open keycharmap file
2012-09-16 22:30:45.993 W 18051/KeyCharacterMap: Error loading keycharmap
file '/system/usr/keychars/SH_touchpanel.kcm.bin'.
hw.keyboards.65541.devname='SH_touchpanel'
```

図 4.2　GPS を利用した際に出力された Logcat ログ

```
11-22 15:33:12.668: I/ActivityManager(280): Starting activity: Intent { act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER] flg=0x10200000 cmp=jp.co.sharp.android.camera/
.stillimagecamera.Camera bnds=[322,320][470,508] }
11-22 15:33:13.064: I/CameraHolder(11474): Camera#open Entry
～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～～
11-22 15:33:13.204: I/ActivityManager(280): Displayed activity jp.co.sharp.android.camera/
.stillimagecamera.Camera: 473 ms (total 473 ms)
11-22 15:33:13.634: I/CameraHolder(11474): Camera#open Exit
```

図 4.3　カメラを利用した際に出力された Logcat ログ

4.4　アプリケーションの特定方法

GPS やカメラの利用を検知する Logcat ログの情報だけでは，それらを利用したアプリケーションを特定することは出来ない．そこで，本提案ではどのアプリケーションが GPS やカメラ等を利用したのかを独自に特定し，提示することとした．

提案方式では，GPS やカメラの利用を Logcat ログにより検知した後，Android 端末で実行中のアプリケーションの名前を取得する．そして，実行中のアプリケーションの中から GPS やカメラの利用を許可するパーミッションが与えられているアプリケーションを調べる．実行中で GPS やカメラの利用を許可するパーミッションが与えられているアプリケーションの内，ユーザ自らインストールしたアプリケーションであるならば，GPS やカメラを不正に利用した可能性のあるアプリケーションとして特定する．

Android ではアプリケーションが自動でインストールされることはない．そのため，マルウェアはユーザが手動でインストールしたアプリケーションの中に存在する可能性が高い．故に，ユーザ自らインストールしたアプリケーションであることをアプリケーションを特定する条件の 1 つとした．

4.5 サーバに蓄積する情報

サーバに蓄積する要注意アプリケーションの情報としては，アプリケーション名，パッケージ名，要求されるパーミッション，パーミッションの組み合わせによって起こりうる被害予想，ユーザが不審に思った動作内容，報告件数，ダウンロード数がある．サーバに蓄積する情報の中では，アプリケーション名，パッケージ名，要求されるパーミッション，ユーザが不審に思った動作内容がユーザの報告により得られる情報である．そして，パーミッションの組み合わせによって起こりうる被害予想，報告件数，ダウンロード数がサーバ側の処理によって得られる情報である．

アプリケーション名とパッケージ名は，ユーザが報告した要注意アプリケーションを一意に識別するために必要である．サーバは，アプリケーション名とパッケージ名により複数のユーザから報告された要注意アプリケーションが同一のものかどうかを判断する．

要求されるパーミッションとパーミッションの組み合わせによって起こりうる被害予想は，要注意アプリケーションが及ぼす被害の可能性を示すために必要である．Android におけるパーミッション機構は，アプリケーションが利用するパーミッション単体の説明をユーザに対して提示するが，パーミッションの組み合わせによって起こる危険性は提示しない．提案方式では，サーバ側で処理したパーミッションの組み合わせによって起こる危険性をユーザに提供することにより，アプリケーションをインストールする際にユーザの安全性の判断を補助する．

ユーザが不審に思った動作内容は，他のユーザが要注意アプリケーションの何に対して不信感を抱いたのか，というアプリケーションの安全性を判断する判断材料となるために必要である．パーミッションの組み合わせによって起こる危険性の提示は，実際に出た被害を提示しているわけではない．この項目により，ユーザは要注意アプリケーションが及ぼす実害を知ることができる．

報告件数とダウンロード数も，ダウンロード数に対してどの程度のユーザが要注意アプリケーションとして報告しているか，というアプリケーションの安全性を判断する判断材料となるために必要である．例えば，1000 件ダウンロードされたアプリケーションであっても要注意アプリケーションとして 500 件報告されていたという情報をユーザに提供することにより，アプリケーションをインストールする際にユーザの安全性の判断を補助する．

提案方式では，Android を利用するユーザの多さを活かして，サーバに蓄積する要注意アプリケーションの情報をより正確にしていく．提案方式を利用することにより，市販のセキュリティ対策ソフトとは異なるマルウェアを判断する基準が確立できる．

第5章　実装

Logcat ログの取得，GPS 利用の検知，カメラ利用の検知，アプリケーションの特定，トーストによるユーザへの通知までの実装を行った．しかし，サーバへ要注意アプリケーションの情報を送信する処理に関しては未実装である．

5.1　実装における課題と解決策

本提案では, Logcat ログを常に取得するアプリケーションを常駐させる必要がある．Android ではサービスというアプリケーションをバックグラウンドで動作させる仕組みがある．本提案はこの仕組みを用いて，Logcat ログを常に取得するアプリケーションを実行させている．しかし，Android ではメモリの不足等により，サービスとして実行中のアプリケーションが OS によって終了させられてしまうことがある．そのため，本提案では Logcat ログが上手く取得できないことがあった．この問題を解決するために， startForeground API[1] を利用した実装を行なった．startForeground API は，サービスとして実行されているアプリケーションをフォアグラウンドとみなして実行させることができる．Android では，startForeground API を利用してフォアグラウンドとして実行されているアプリケーションは，通常のバックグラウンドで実行されるプロセスよりも優先度が高くなり，強制終了の可能性が低くなる [20].

5.2　ビジュアル化プログラムの構成

図 5.1 に Android アプリケーションとして実装したビジュアル化プログラムの構成を示す．MainActivity クラスのメソッドの機能は以下の通りである．

- onCreate メソッド
  クラスの中で最初に実行されるメソッド．サービス処理の呼び出しを行っている．

サービスで行う処理をまとめた PlayerService クラスのメソッドの機能は以下の通りである．

- onCreate メソッド
  クラスの中で最初に実行されるメソッド．Logcat ログを取得する処理の呼び出しを行う．また，ノーティフィケーションの処理の呼び出しを行う．

[1] startForeground API を利用するには，Android 端末のステータスバーにアプリケーションが実行中であるという通知を表示する制約がある．

- showNotification メソッド
  端末のステータスバーにアイコンやメッセージを表示する．また，サービスをフォアグラウンドとみなして実行する．

- getLog メソッド
  Logcat ログの取得を行う．また，アプリケーション挙動の検知処理の呼び出しやトースト表示処理の呼び出しを行う．

- showToast メソッド
  挙動を検知したアプリケーションの特定を行う．また，トーストの表示を行う．

- collation メソッド
  シグネチャと取得ログの照合を行う．

本提案が実装された Android 4.0 を搭載した HTC Corporation の ISW13HT で，GPS とカメラを利用したところ，それらの機能の利用を知らせるトーストがアプリケーション名と共に表示されることが確認できた．

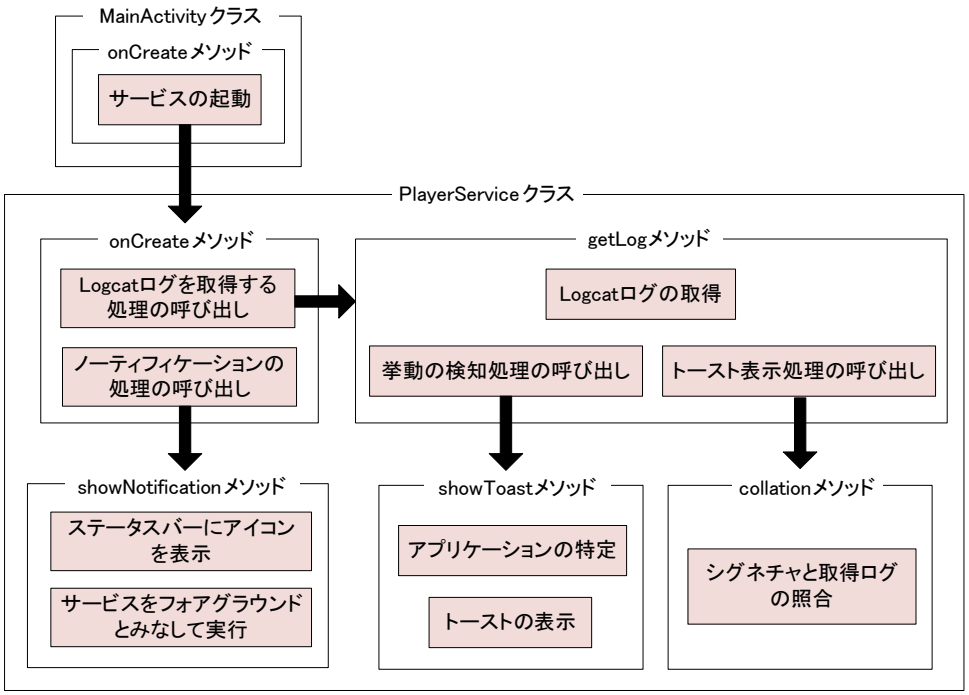

図 5.1　ビジュアル化プログラムの構成

5.3 実装における検証

本項では，startForeground API を利用した実装における工夫の有効性を示す．以下の 2 つの場合において，本提案で用いるアプリケーションを実行させ，実装における工夫の有効性を確認した．

1. 対策を施さなかった場合

2. startForeground API を利用した実装を行った場合

対策を施さなかった場合と startForeground API を利用した実装を行った場合において，アプリケーションがバックグラウンドで正常に動作して Logcat ログを取得していることと，Android 端末のバッテリ残量の推移状況を確認した．調査に利用した Android 端末は，Android4.0 を搭載した HTC Corporation の ISW13HT である．1 時間に 1 回，特定のログを出力するアプリケーションを作成して，Logcat ログを常時取得するビジュアル化プログラムと同時実行した時に，特定のログを取得できるか否かの確認を行った．同時実行の期間は 1 日とし，対策を施さなかった場合と startForeground API を利用した実装を行った場合，それぞれ 5 回確認を行った．この Logcat ログの取得状況を，アプリケーションが OS に強制終了させられずに正常に実行されているかどうかの目安とした．表 5.1 に，1 日の間に出力される Logcat ログ 25 個中[2]，取得できた Logcat ログの数と Logcat ログの取得率を示す．表 5.1 に示す値は，取得できた Logcat ログの数と Logcat ログの取得率の 5 回分の平均値である．

表 5.1 より，対策を施さなかった場合の Logcat ログ取得率が 56.8%であったのに対し，startForeground API を利用した実装を行った場合が 93.6%と高い取得率であった．この結果から，startForeground API を利用した実装を行った場合が，アプリケーションを常駐させていても OS に強制終了されにくいことが確認できた．取得すべき Logcat ログの個数が増えた場合においても，ログ取得率に関しては変化しないものであると考える．

また，図 5.2 より，対策を施さなかった場合のバッテリ残量は，95.2%であるのに対し，startForeground API を利用した実装を行った場合は，91.8%であった．図 5.2 に示す値は，5 回確認したバッテリ残量の推移の平均値である．対策を施さなかった場合について，バッテリ残量の減少が止まっている部分があることがわかる．これは，対策を施さなかった場合のログ取得率が低いことからも，本提案で用いるビジュアル化プログラムがたびたび強制終了させられ，常に実行されている状態になかったことが考えられる．また，startForeground API を利用した実装を行った場合では，バッテリ残量の減少は約 10%以内に止めている．さらに，表 5.1 を見てもログ取得率が 93.6%であることから，本提案で用いるビジュアル化プログラムは OS による強制終了を避けつつ常駐できていると考えられる．この結果から，startForeground API を利用した実装の有効性が確認できた．

[2]実行直後にも Logcat ログを出力するため 25 個になる．

表 5.1　Logcat ログの取得率

	ログ取得成功数	ログ取得率
対策なし	14.2 個	56.8%
startForeground API による対策	23.4 個	93.6%

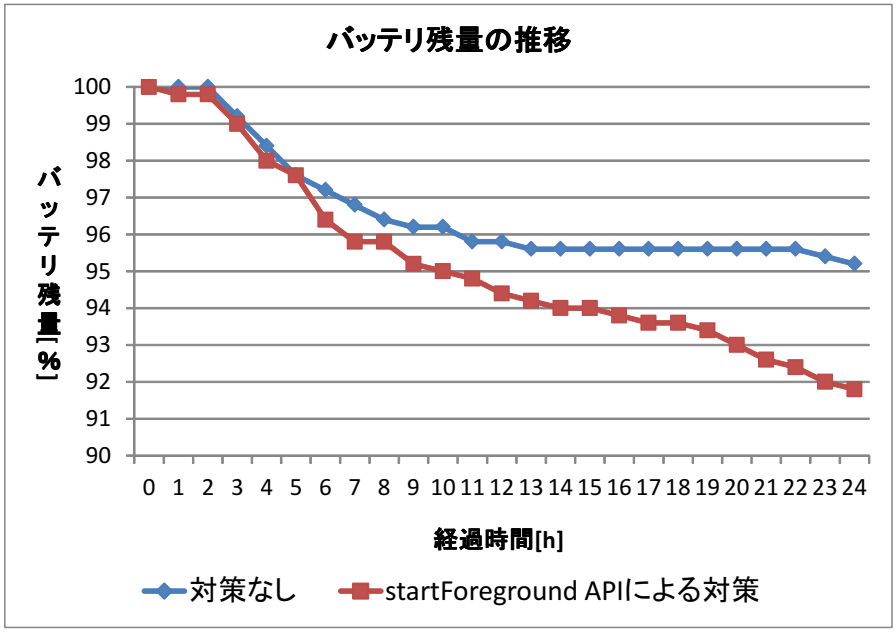

図 5.2　バッテリ残量の推移

第6章　評価

6.1　市販のセキュリティ対策ソフトとの比較

本研究を進めるにあたり，市販のセキュリティ対策ソフト[1]，提案方式，市販のセキュリティ対策ソフトと提案方式を併用した3つの場合について比較を行った．比較は，2.1節でパーミッション機構の課題として述べた，アプリケーションインストール前後における対応の有無，システム導入のし易さ，端末操作の快適性の維持，新種・亜種のマルウェアへの対応の5項目について行った．表6.1にその結果を示す．また，提案方式はサーバ部分が未実装であるため，サーバとの通信に関するオーバーヘッドやサーバの維持にかかるコストについては，評価に反映していない．

まず，市販のセキュリティ対策ソフトについては，アプリケーションの更新やインストール時に自動スキャンを行うため，インストール前のマルウェア対策は行われている．そのため「○」とした．インストール後のマルウェア対策としては，インストールされているアプリケーションのスキャンを行うが，アプリケーションの保存場所によってセキュリティ対策ソフトのスキャンが及ばない可能性がある．そのため「△」とした．また，システム導入のし易さについては，Android端末にセキュリティ対策ソフトをインストールすることで適用されるため，システムの導入は容易である．そのため「○」とした．さらに，アプリケーションインストール時の自動スキャンなどユーザに不自由を与えることなくマルウェア対策が行われているため，快適性は維持されていると考え，「○」とした．新種・亜種のマルウェア対策は，ヒューリスティック検知を利用した未知マルウェアの検出も行っている．しかし，ヒューリスティックによるマルウェア検知であってもマルウェアの誤検知の可能性や検出できない可能性も含んでいる．そのため「△」とした．

次に提案方式については，ユーザはサーバに蓄積された要注意アプリケーションの情報を参考にしてアプリケーションのインストールの可否を決めることができるため，インストール前のマルウェア対策は行われている．しかし，正当性の判断はユーザに一任され，必ずしもユーザが正しい判断をするとは限らないため「△」とした．インストール後のマルウェア対策としては，アプリケーションの挙動を可視化してユーザに通知するため，ユーザは容易に不審なアプリケーションを見つけてアンインストールすることができる．しかし，インストールされたマルウェアによる被害を防ぐことはできない．そのため「△」とした．また，システム導入のし易さについては，Androidのアプリケーションであるビジュアル化プログ

[1]市販のセキュリティ対策ソフトとして想定しているのは，ノートンモバイルセキュリティである．

ラムを Android 端末にインストールすることで適用されるため，システムの導入は容易である．そのため「○」とした．アプリケーション挙動の可視化については，鬱陶しく感じるユーザもいると思うが，通知方法をトーストによる一時的な表示にすることにより，ある程度の快適性を維持している．そのため「△」とした．さらに，新種・亜種のマルウェア対策は，アプリケーションの挙動を可視化することにより，市販のセキュリティ対策ソフトで検出できなかった新種・亜種のマルウェアをユーザが見つけることができる可能性がある．そして，世界中の Android 端末ユーザから寄せられた要注意アプリケーションの情報を参考にすることにより，市販のセキュリティ対策ソフトで検出できなかった新種・亜種のマルウェアをインストールの段階で防ぐことができる可能性がある．しかし，正当性の判断はユーザに一任されるため，必ずしもユーザが正しい判断をするとは限らない．そのため「△」とした．

最後に市販のセキュリティ対策ソフトと提案方式を併用した場合については，セキュリティ対策ソフトの自動スキャンと本提案のサーバに蓄積された情報を利用したユーザによるアプリケーションの正当性の判断により，個別でのマルウェア対策よりも厳重なマルウェア対策がインストール前に行われている．そのため「○」とした．インストール後のマルウェア対策としては，セキュリティ対策ソフトのスキャンが及ばないマルウェアの挙動も本提案では可視化され，ユーザに通知するため，不審なアプリケーションを見つけてアンインストールすることができる．そのため「○」とした．また，システム導入のし易さについては，両者とも Android 端末にアプリケーションとしてインストールすることにより，マルウェア対策が適用されるため，システムの導入は容易である．そのため「○」とした．市販のセキュリティ対策ソフトにおいては端末の快適性は維持されるが，アプリケーション挙動の可視化については，鬱陶しく感じるユーザもいる可能性がある．そのため「△」とした．さらに，新種・亜種のマルウェア対策は，セキュリティ対策ソフトで検出できなかったマルウェアを本提案で見つけることができる可能性があり，本提案で見つけることができなかったマルウェアをセキュリティ対策ソフトで見つけることができる可能性がある．お互いの良さをとることにより，個別でのマルウェア対策よりも新種・亜種のマルウェア対策として有効であると考え，「○」とした．

表 6.1　市販のセキュリティ対策ソフトと本提案の比較

	市販のセキュリティ対策ソフト	提案方式	両者の併用
課題 1 への対応	○	△	○
課題 2 への対応	△	△	○
導入のし易さ	○	○	○
快適性の維持	○	△	△
新種・亜種への対応	△	△	○

6.2 関連研究との比較

本研究を進めるにあたり，特に参考にした 3 つの関連研究と本提案の比較を行った．表 6.2 に，提案方式，マーケットでマルウェア対策を行っている文献 [5] の方式，インストール前にアプリケーションの危険性を示してくれる文献 [7] の方式，API ごとに機能利用のリアルタイム動的制御を行う文献 [9] の方式の比較結果を示す．比較は，2.1 節でパーミッション機構の課題として述べたインストール前後のマルウェア対策の有無，システム導入のし易さ，端末操作の快適性の維持，新種・亜種のマルウェアへの対応の 5 項目について行った．

文献 [5] は，アプリケーションをマーケットへ公開する前に審査を行う．これにより，端末へのマルウェアの感染を防ぐことができるため，インストール前のマルウェア対策は行われている．そのため「○」とした．しかし，審査を通過したマルウェアに関しては対処することができないため，インストール後のマルウェア対策は行われていない．そのため「×」とした．事前審査ツールは Android アプリケーションとして実装されているが，マーケットにアプリケーションの事前審査の仕組みを適用することは容易ではない．そのため「△」とした．また，アプリケーションの事前審査以外は特別なことは行わないため，ユーザが端末を利用する際の快適性は維持されている．そのため「○」とした．さらに，新種・亜種のマルウェアの対応は，審査を行うのは評価者であり，確実にに対応できるとは限らないため「△」とした．

文献 [7] は，インストール前にアプリケーションの危険性をユーザに提示することにより，マルウェアの感染を防ぐため，端末へのマルウェアの感染を防ぐことはできる．そのため「○」とした．しかし，一旦インストールしてしまったマルウェアに関しては対処することができないため，インストール後のマルウェア対策としては行われていない．そのため「×」とした．危険性をユーザに提示する Android アプリケーションをインストールすることがシステムの導入となるため，システムの導入は容易である．そのため「○」とした．また，インストール時に危険性をユーザに提示するのみの対策であるため，端末操作の快適性は維持される．そのため「○」とした．さらに，インストール時のマルウェアの検知率は高いが，インストール後の対応がないため，新種・亜種のマルウェアへの対応については「△」とした．

文献 [9] は，アプリケーションがパーミッションに保護されている機能を利用しようとする度に，ユーザへ機能利用の承認を求める仕組みのため，インストール前のマルウェア対策は行われていない．そのため「×」とした．また，マルウェアによる不審な挙動をユーザ自身で止めることができるため，インストール後のマルウェア対策は行われている．そのため「○」とした．このシステムは Application Framework で実装されているので，Android 自体を改造しなければならず，システムの導入は難しい．そのため「×」とした．また，機能利用の承認を求められることに鬱陶しさを感じるユーザもいると考えられる．しかし，一度承認すれば再度承認を求めない設定ができるため，端末操作の快適性の維持には配慮している．そのため「○」とした．さらに，API レベルでアプリケーション挙動のフックを行っており，新種・亜種のマルウェアの不審な挙動についても対応できる．しかし，アプリケー

ションによる機能の利用を最終的に許可するのはユーザであるため「△」とした．

まず，表 6.2 における評価項目のうちの，アプリケーションのインストール前後でマルウェア対策が行われているかについて考察する．マーケットでマルウェア対策を行っている文献 [5] の方式と，インストール前にアプリケーションの危険性を示してくれる文献 [7] の方式は，アプリケーションインストール後においてマルウェアの被害拡大を防ぐ手段がない．また，API ごとに機能利用のリアルタイム動的制御を行う文献 [9] の方式は，アプリケーションのインストール前にマルウェアの感染を防ぐ手段がない．対して，提案方式はアプリケーションのインストール前後でマルウェア対策を行っている．提案方式ではサーバに蓄積された要注意アプリケーションの情報を確認した後，そのアプリケーションをインストールするか否かはユーザに一任される．また，アプリケーションの挙動をユーザに通知した後，そのアプリケーションの正当性を判断して削除するかどうかを決めることもユーザに一任される．そのため，提案方式ではアプリケーションのインストール前後で確実にマルウェアの感染を防止することはできないため，「△」と評価している．しかし，セキュリティの分野では，複数のセキュリティ対策を組み合わせてセキュリティ性を向上させていくことを考える．提案方式と市販のセキュリティ対策ソフトを併用した場合は，提案方式がアプリケーションのインストール前後でマルウェア対策を行っている分，アプリケーションのインストール前後のどちらか一方しかマルウェア対策を行っていない関連研究と市販のセキュリティ対策ソフトを併用した場合よりもセキュリティ性が高くなることが考えられる．

次に，システム導入のし易さについて考察する．マーケットでマルウェア対策を行っている文献 [5] の方式は，マーケットにアプリケーションを事前審査する仕組みを適用する必要があるため，システムの導入は難しい．また，API ごとに機能利用のリアルタイム動的制御を行う文献 [9] の方式は，Application Framework を改造する必要があるため一般の Android 端末に容易には適用できない．対して，提案方式とインストール前にアプリケーションの危険性を示してくれる文献 [7] の方式は，独自のアプリケーションを Android 端末にインストールするだけでシステムが適用されるため，システム導入は容易である．

さらに，快適性の維持について考察する．表 6.2 で挙げた全ての関連研究では快適性が維持されている一方で，提案方式はアプリケーション挙動のトースト表示に鬱陶しさを感じるユーザがいることを踏まえて「△」と評価している．しかし，提案方式も API ごとに機能利用のリアルタイム動的制御を行う文献 [9] の方式のように，今後ユーザが一度承認すればアプリケーションの挙動を通知しない設定をする実装ができれば端末操作の快適性の維持についても実現できると考える．

最後に，新種・亜種のマルウェアへの対応について考察する．提案方式と表 6.2 で挙げた関連研究は，全て「△」と評価している．そのため，提案方式と表 6.2 で挙げた関連研究において，市販のセキュリティ対策ソフトと併用したことを考えると全ての方式でセキュリティ性の向上が期待でき，提案方式と表 6.2 で挙げた関連研究における優劣はつかないものであると考える．

表 6.2　関連研究と本提案の比較

	提案方式	マーケットでの事前審査	危険性提示	機能利用の動的制御
課題 1 への対応	△	○	○	×
課題 2 への対応	△	×	×	○
導入のし易さ	○	△	○	×
快適性の維持	△	○	○	○
新種・亜種への対応	△	△	△	△

第7章　まとめ

本論文では，Android 端末において，ユーザにおけるマルウェアの発見やアプリケーションインストール時の安全性の判断を補助するシステムを提案した．提案方式では，Logcat ログから GPS 等の機能の利用や個人情報の外部サーバへの送信を検知し，その挙動をトーストで可視化する．その情報から，ユーザ自身に要注意アプリケーションかどうかの判断を行わせ，要注意と判断したならばアンインストールを行う．さらに，ユーザはサーバに要注意アプリケーションの情報を報告し，それを蓄積する．蓄積した情報をユーザ間で共有することにより，新たにインストールするアプリケーションの正当性を判断するユーザを補助する．提案方式により，ユーザのアプリケーションインストール時の安全性の判断を補助し，インストール後も挙動の可視化により不審なアプリケーションの存在をユーザに気付かせることを可能とした．

本研究では提案方式の実装を行い，GPS やカメラを利用した際にユーザに機能の利用を知らせるトーストが表示されることを確認した．さらに，提案方式の評価として，提案方式と市販のセキュリティ対策ソフトを個別に使用した場合及び，両者を併用した場合において定性評価を行った．さらに，提案方式と関連研究においても定性評価を行い，提案方式の有効性を確認した．

今後の課題としては，Logcact ログ以外でのアプリケーション挙動の検知方法を検討しつつ，未実装であるサーバ部分の実装を行うことである．

謝辞

本研究を遂行するにあたり，多大なるご指導そしてご協力を頂きました，名城大学大学院理工学研究科　渡邊晃教授に心より厚く御礼申し上げます.

また，本論文を制作するにあたり，多大なるご指導そしてご協力をいただきました，名城大学大学院理工学研究科　吉川雅弥教授，鈴木秀和助教，旭健作助教に心より厚く御礼申し上げます.

本研究を遂行するにあたり，有益なご助言，適切なご検討をいただきました，渡邊研究室，鈴木研究室の皆様に心より感謝致します.

最後に，私の研究生活をいつも暖かく支えてくれた家族に心より感謝致します.

参考文献

[1] ITmedia, Android 世界スマートフォン総出荷で 8 割超 - IDC 調べ
http://www.itmedia.co.jp/mobile/articles/1311/13/news071.html

[2] McAfee 脅威レポート_2013 年第 3 四半期
http://b2b-download.mcafee.com/products/japan/pdf/threatreport/threatreport13q3.pdf

[3] Android Developers -System Permissions-
http://developer.android.com/guide/topics/security/permissions.html

[4] ITSCJ，スマートフォンセキュリティの課題と国際標準化
http://kikaku.itscj.ipsj.or.jp/topics/nl94_smartphone.html

[5] 竹森敬祐，磯原隆将，窪田歩，高野智秋：Android 携帯電話上での情報漏洩検知，
The 2011 Symposium on Cryptography and Information Security Kokura, Japan, Jan.2011.

[6] 竹森敬祐，川端秀明，磯原隆将，窪田歩，池野潤一：Android フォンのアプリ管理〜ホワイトリスト方式〜， Vol.2011-CSEC-53 No.2, May.2011.

[7] 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，Vol.2012-CSEC-56 No.12, Feb.2012.

[8] Mohammad Nauman, Shail Khan, Xinwen Zhang: Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints, ACM Symposium on Information, Computer and Communications Security, 2009.

[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 における細粒度アクセス制御機構，情報処理学会論文誌，Vol.54 No.8, Aug.2013.

[10] 上松晴信，可児潤也，米山裕太，川端秀明，磯原隆将，竹森敬祐，西垣正勝：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（その 2），Vol.2013-CSEC-60 No.57, March.2013.

[11] 于競，山内利宏：Android における WebView の脆弱性を利用した攻撃を防止するアクセス制御方式の提案，Vol.2013-CSEC-60 No.4, March.2013.

[12] 葛野弘樹：Android アプリケーションに対する情報フロー制御機構の提案，Computer Security Symposium 2011,Oct.2011.

[13] William Enck, Peter Gilbert, Byung-Gon Chun: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, 9th USENIX Symposium on Operating Systems Design and Implementation, 2010.

[14] 名雲孝昭，秋山満昭，針生剛男：ContentProvider を用いた利用者情報送信の動的解析手法に関する検討，Vol.2013-CSEC-60 No.52, March.2013

[15] ノートンモバイルセキュリティ
http://jp.norton.com/productDetails.do?fm=NMS&ver=V2.0

[16] 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，発行人：土田米一，発行：株式会社インプレスジャパン，発売：株式会社インプレスコミュニケーション

[17] 最新ウイルス一覧
http://www.mcafee.com/japan/security/latest.asp

[18] 2013 年ノートンレポート
http://www.symantec.com/ja/jp/about/news/resources/press_kits/detail.jsp?pkid=norton-report-2013

[19] ITpro, Android の仕組みを知る (1)
http://itpro.nikkeibp.co.jp/article/COLUMN/20091126/341182/?ST=android-dev&P=2

[20] Android Developers -Service-
http://developer.android.com/reference/android/app/Service.html

研究業績

学術論文

なし

研究会・大会等

1. 戸田尚希, 鈴木秀和，渡邊晃：Android 端末をターゲットとしたボットによる被害防止策の検討，平成 23 年度電気関係学会東海支部連合大会論文集，F1-4，2011.

2. 戸田尚希, 鈴木秀和，渡邊晃：Android 端末をターゲットとしたボットによる被害防止策の提案，情報処理学会第 74 回全国大会講演論文集，Mar.2012.

3. 戸田尚希, 鈴木秀和，旭健作，渡邊晃：Android アプリケーションの挙動を可視化することによるセキュリティ対策，情報学ワークショップ 2012（WiNF2012）論文集，WiNF2012，Vol.2012，pp.115-118，Dec.2012.

4. 戸田尚希, 鈴木秀和，旭健作，渡邊晃：Android アプリケーションの挙動を可視化することによるセキュリティ対策の検討，マルチメディア, 分散, 協調とモバイル (DICOMO2013) シンポジウム論文集，Vol.2013，No.1，pp.1348-1354，Jul.2013.