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

戸田 尚希[1] 鈴木 秀和[1] 旭 健作[1] 渡邊 晃[1]

概要：Android 端末の普及に伴って，Android をターゲットにしたマルウェアが数多く確認されるようになっている．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．本稿では，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，ユーザが危険であると判断した場合はアンインストールを促す．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．

Consideration of Security Measures to Visualize the Behavior of Android Applications

Toda Naoki[1] Suzuki Hidekazu[1] Asahi Kensaku[1] Watanabe Akira[1]

1. はじめに

インターネットの発展に伴い，誰でも自由にインターネットを利用できるようになった．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．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 [1]．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．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てはじめて，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．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，感染する恐れは軽減できる．しかし，ユーザが常にパーミッションを確実にチェックするとは限らない．そのため，感染してしまった後についてもそのことを検出し，対策をとる手段は必要である．

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 [4], [5], [6] と，感染後の被害を抑える方式 [7], [8] に大別できるが，本提案では両者に対応していることが特

[1] 名城大学大学院理工学研究科情報工学専攻

徴である．本稿では，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．提案システムは，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補助する．

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．5 章でまとめる．

2. Android セキュリティの課題

2.1 マルウェア被害の分類

表 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．これは，文献 [2] 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，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，端末内部の設定を変更する被害である．ダウンロード被害はアプリケーションを勝手にダウンロードする被害である．ダウンロード被害は，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．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 [2]．

この 2 種類の被害を比較すると，ユーザにとって被害が大きいのは情報送信被害である．その理由は，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からである．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．そのため，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 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
他の被害	送られてくる SMS メッセージの監視
	指定した Web サイトにアクセス
	ルート権限の取得

2.2 パーミッション機構の課題

Android で採用されてい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．

- 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 1）．
- インストールしたマルウェアに対しては被害の拡大を防ぐことができない（課題 2）．

Android では，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がユーザに表示される．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も一番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，ユーザがその表示から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．そのため，パーミッション機構におけるセキュリティ対策は十分に効力を発揮していない状況にある．

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．

2.3 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．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 [3]．

図 1 Android のセキュリティ対策における関連研究

3. 関連研究

図 1 に Android のセキュリティに関連する研究の一覧を示す．図 1 は，これらの技術が，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．

3.1 Market における対策

文献 [4] 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．この提案では，Logcat ログに注目している．Logcat ログとは，アプリケーションやシステムから出力されるログである．取得した 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．

3.2 Application 層における対策

文献 [5] 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．この提案では，法人管理者が予めアプリケーションを実行して，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，脅威の低いアプリケーションのホワイトリストを作成する．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．また，アプリケーションが認定されていない場合，ユーザ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 [6] 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．この提案では，パーミッションの組み合わせに着目している．特定のパーミッションの組み合わせを持つアプリケーションでは，位置情報の流出等の危険がある．そのため，位置情報の流出等の危険性を示すアイコンを表示するユーザインタフェースを作成している．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判断を補助する．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，マルウェアでない Android 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．したがって，この提案は，誤検知率の高さが課題である．

3.3 Application Framework 層における対策

文献 [7] 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，拡張した Android 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．セキュリティポリシーの設定によって個人情報や位置情報へのアクセスを許可したり，拒否したりすることができる．しかし，API が呼び出されるたびにセキュリティポリシーを参照するため，個人情報等への参照に時間がかかるという課題がある．

文献 [9] 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．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．

3.4 Linux Kernel 層における対策

文献 [8] 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である．しかし，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．

4. 提案方式

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が，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，情報の漏洩等を引き起こす．この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行う方式とした．本提案は，アプリケーション層での実装が可能であり，Android のバージョンや機種に関わらず，適用が可能である．

4.1 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可視化する．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 2 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定義ログとして予め定義する．Android には，アプリケーションやシステムのログをリングバッファに一時的に保存する機能がある．このバッファに保存されたログを，Logcat コマンドを用いて参照する．この際，参照したログが定義ログと一致する場合，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，その旨を簡単なメッセージを一時的にユーザに表示する機能であるトーストにて表示する．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，不正な挙動の内容とその動作をしたアプリケーション名を表示する．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．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．

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．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．

本提案ではユーザ自身がトーストで表示された内容を確認して，その表示内容を適切であるか，不適切であるかを判断しなければならない．そのため，ユーザのセキュリティ意識を高めることも期待できる．

現在は，Logcat ログの取得，GPS 利用の検知，トーストによるユーザへの通知，アプリケーションの特定までの実装が完了している．しかし，サーバへ要注意アプリケーション情報を送信する処理に関しては未実装である．

4.2 Logcat ログの調査

本提案では Logcat ログから GPS 等の機能の利用や個人情報の送信などを検知する．今回は GPS 利用時にどのようなログが出力されるのかを調査し，提案方式における定義ログを決定した．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．

今回，Logcat ログの確認をした実機は Android2.2 を搭

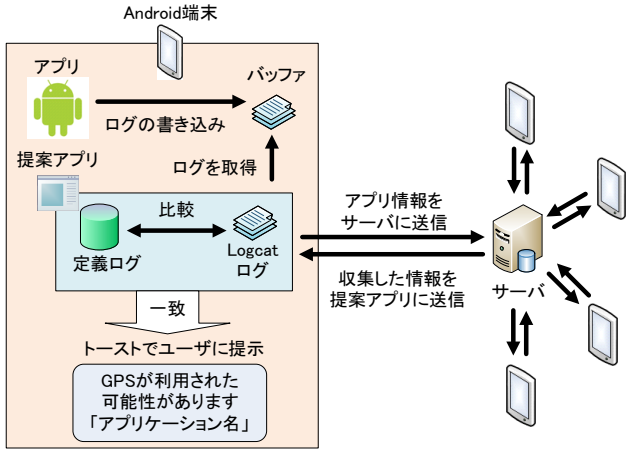

図 2 提案方式の動作概要

```
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'
2012-09-16 22:30:46.003 W 18051/KeyCharacterMap: Using default keymap: /
system/usr/keychars/qwerty.kcm.bin
```

図 3　GPS 利用時の Logcat ログ

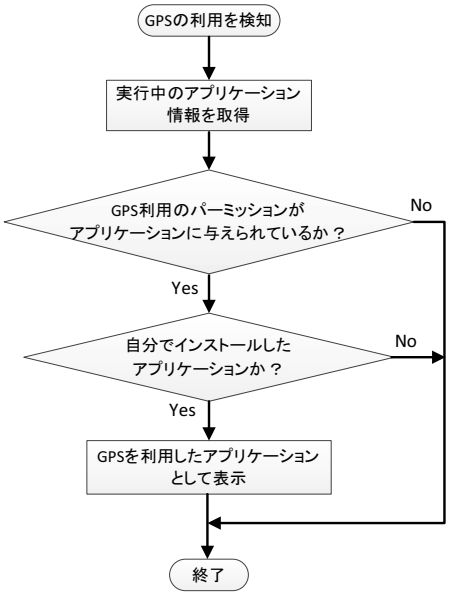

図 4　GPS を利用したアプリケーションを特定するフローチャート

載したシャープ (株) の IS03 である．図 3 は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．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 に関するログを定義ログとして利用できると考えられる．

4.3　マルウェアの特定

GPS の利用を検知する Logcat ログの情報だけでは，GPS 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，本提案ではマルウェアの可能性のあるアプリケーションを独自に特定し，提示することとした．

図 4 に，GPS を利用した可能性のあるアプリケーションを特定するまでのフローを示す．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．GPS の利用を Logcat ログにより検知した時，現在実行中のアプリケーション情報を取得する．実行中アプリケーション情報の中から，GPS を利用する際に必要なパーミッションが与えられており，かつ，そのアプリケーションをユーザが自身でインストールしたものである場合に GPS を利用したアプリケーションとして特定する．

4.4　実装における検証

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．本提案では定期的にアプリケーションを起動することにより OS によってアプリケーションが終了させられたとしても，Logcat ログが取得できなくなる状況を防ぐこととした．具体的には，Android の AlarmManager という仕組みを利用して，Logcat ログを取得する処理部分を定期的に呼び出す．

Logcat ログを取得する処理部分の定期的な呼び出しは，端末のバッテリ消費が著しく増加する可能性が考えられる．そのため，今回は，呼び出し間隔を 30 分，15 分，1 分，呼び出しなし，とした 4 つの場合において，Logcat ログを常時取得するアプリケーションを実行し，端末のバッテリ残量の変化を確認した．確認に利用した 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，Logcat ログを常時取得するアプリケーションにて特定のログを取得できるか否かの確認を行った．表 2

表 2　Logcat ログの取得率

	ログ取得成功数	ログ取得率
呼び出しなし	10 個	40%
30 分間隔	23 個	92%
15 分間隔	24 個	96%
1 分間隔	25 個	100%

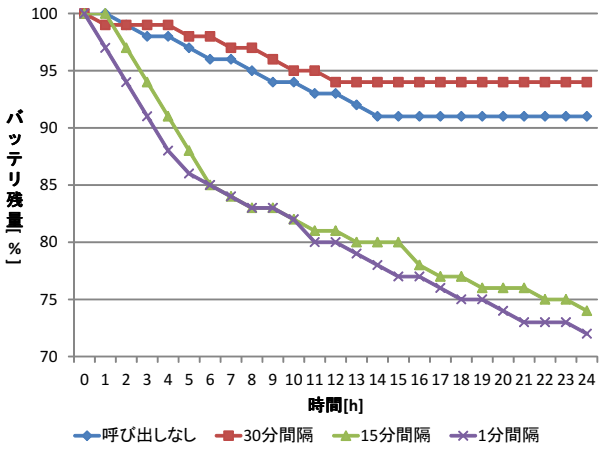

図 5　Android 端末におけるバッテリ残量の変化

表 3　関連研究との比較

	提案方式	文献 [4]	文献 [6]	文献 [7]
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	△	○	○	△

に，1 日の間に出力される Logcat ログ 25 個中，取得できた Logcat ログの数と Logcat ログの取得率を示す．

アプリケーションの定期的な呼び出しを行わなかった場合，10 個の Logcat ログしか取得できなかったのに対し，呼び出し間隔を 30 分，15 分，1 分とした場合，それぞれ 23 個，24 個，25 個のログが取得できた．このことから，Logcat ログを常時取得するアプリケーションを定期的に呼び出す方法は有効であることがわかる．

図 5 は，Logcat ログを常時取得するアプリケーションを定期的に呼び出した場合の，バッテリ残量の変化について調査した結果である．30 分間隔で呼び出しを行った場合，調査開始から 24 時間後のバッテリ残量は 90%を越えていることが確認できた．呼び出しなしの場合とバッテリ残量に大差がないため，30 分間隔での呼び出しはバッテリ消費の面からも問題ないものであることがわかった．それに対して，15 分間隔と 1 分間隔での呼び出しの場合，両者ともバッテリ残量は 75%を下回った．端末に対して操作を行っていないにもかかわらず，1 日でバッテリ残量が 75%を下回ることは望ましくない．そのため，30 分間隔で Logcat ログを常時取得するアプリケーションの呼び出しを行なうこととした．

4.5　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 つの関連研究と提案方式の比較を行った．表 3 に，提案方式，文献 [4] の方式，文献 [6] の方式，文献 [7] の方式の比較結果を示す．比較は，2.2 節で述べた課題 1，課題 2 への対応の有無，システム導入のし易さ，端末操作の快適性の維持の 4 項目について行った．

提案方式では，ユーザから報告された要注意アプリケーションの情報をアプリケーションのインストール前に確認するため，マルウェアの感染を防ぐことができる．また，アプリケーションの挙動をトーストを用いて可視化するため，マルウェアが端末に感染してユーザの意図しない動作を行ったとしても，ユーザによるマルウェアの発見を補助できる．そして，アプリケーションのアンインストールを促すことによりマルウェア被害の拡大を防ぐことができる．さらに，提案方式はアプリケーション層でのマルウェア対策であるため，システムの導入は容易である．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で，ある程度の快適性を維持している．

文献 [4] では，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ため，端末へのマルウェアの感染を防ぐことはできる．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．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．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．

文献 [6] では，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．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．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．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．

文献 [7] では，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ため，マルウェアの感染を防ぐことはできないが，マルウェアによるアプリケーションの挙動をユーザ自身で止めることができる．このシステムは Application Framework 層で実装されているので，端末自体を改造しなければならず，システムの導入は難しい．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．

5.　まとめ

本提案では，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，ユーザはサーバに報告し，その情報を蓄積する．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．このため，マルウェアをインストール前に防ぐことができる可能性がある．今後は，GPS 利用以外の定義ログを決定するために有用なログの調査をしつつ，提案方式の有効性を確認するための実装を進める予定である．

参考文献

[1] McAfee 脅威レポート，2012 年第 2 四半期入手先 ⟨http://www.mcafee.com/japan/media/mcafeeb2b/international/japan/pdf/threatreport/threatreport12q2.pdf⟩

[2] McAfee 最新ウイルス一覧入手先 ⟨http://www.mcafee.com/japan/security/latest.asp⟩

[3] 齋藤長行，吉田智彦，”青少年のスマートフォン利用環境整備のための政策的課題－実証データ分析から導かれる政策的課題の検討－”入手先 ⟨http://www.mcafee.com/japan/security/latest.asp⟩

[4] 竹森，他，”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.

[5] 竹森，他，“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.

[6] 松戸，他,”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.

[7] 川端，他，“Android OS における機能や情報へのアクセス制御機構の提案”, Computer Security Symposium 2011, Oct.2011.

[8] 葛野，“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.

[9] Mohammad, other, “Apex:Extending Android Permission Model and Enforcement with User-defined Runtime Constraints”, ACM Symposium on Information, Computer and Communications Security, 2009.

[10] 上松，他，“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”, Vol.2012-CSEC-57 No.18, May.2012.