Vol.2013-CSEC-61 No.6
Vol.2013-IOT-21 No.6
2013/5/9

ユーザの承認によるアプリケーション実行時の Android マルウェア対策

加藤真† 松浦佐江子†

Android マルウェアは増加傾向にあり，その中で特に，人気ゲームなどの名前を騙りユーザに許可されたアプリケーションの動作に対する権限の範囲で悪事を働くものが氾濫している．そのようなマルウェアは権限確認で感染を防ぐことが出来るが，権限によってアクセスが制限されるべき機能に権限が付加されていない・インテントの利用によりアプリケーションに許可された権限以上のことができる等の権限システムの問題があり，権限確認だけではユーザがマルウェアを正しく認識するには万全ではない．本研究ではアプリケーションが権限を要求する動作を実行する時に，それが不正かつ有害な動作でないか判断できる情報をユーザに提示し，動的に承認を行うシステムを提案する．これによりユーザは，アプリケーションのユーザ情報へのアクセスや機能の使用を細かくコントロールすることが可能になる．

A Dynamic Countermeasure Method to Android Malware by User Approval

MAKOTO KATO† SAEKO MATSUURA†

Android malware is increasing and flooded with malware which disguise as popular applications and execute malicious attacks within the scope of permissions which was approved by users. Such malware can be prevented by confirming all risky permissions. However confirming permissions is not full preparation for preventing infection, because of defective Android Permission System. Several functions which should be restricted are often not restricted by appropriate permissions. Moreover, applications can go beyond the approved permission by using Intent. This paper proposes a dynamic approval system. When an application executes an action which requires the permission, the proposed system shows a user such adequate information that he/she can judge whether the application is malicious or not. So he/she can determine whether to grant an application's action. The aim of the proposal system is to make fine grained application controllability, so that users can protect their personal valuable information from malware.

1. はじめに

近年，急速に普及しているスマートフォンは電話番号やメールアドレス，位置情報など個人を特定可能な情報を多く持つ．スマートフォン向け OS（Operating System）の一つである Android[1]は，オープンソースで，アプリケーションのマーケットに登録する時に審査がないなど，iOS や Windows Phone などの他のスマートフォン OS とは異なりオープンな仕様を持つ．これらの特徴と Android を搭載したスマートフォンは市場トップシェアであることから，Android は格好の攻撃対象であり Android を対象とした不正かつ有害な動作をする悪質なコード群であるマルウェアは増加している．

1.1 Android マルウェア

Android におけるマルウェアには，ユーザに許可された権限の範囲で悪事を働くものがある．このマルウェアに感染するかどうかは，アプリケーションインストール時に表示されるそのアプリケーションの動作に必要な権限をユーザが確認するかどうかに依存する．よって，ユーザは自身の判断で権限の確認を行うことによって必要以上の権限を要求するアプリケーションをマルウェアと見抜くことができ，感染を防ぐことが出来る．

1.2 Android のセキュリティシステム

Android は次の 2 つのセキュリティシステムによって保護されている．

i) サンドボックス
ii) 権限システム

Android のアプリケーションは，サンドボックスと呼ばれる独立した，且つ保護された領域で動作しており，これによりアプリケーションから不正にシステムを操作されたり，他のアプリケーションのリソースにアクセスされたりするのを防いでいる．コンピュータ向けの OS である Linux では，UID（User Identifier）や GID（Group Identifier）を用いて，ユーザをグループに所属させたり，各ユーザ，各グループのリソースを分離し，リソースへのアクセス制限を行っている．Android は Linux をベースにした OS のため，この仕組みを流用している．

アプリケーションをインストールすると，システムによってアプリケーションにユニークな UID とそのアプリケーションが要求する権限に基づいた GID がアプリケーションに割り当てられる．この割り当てられた UID を基に各アプリケーションのリソースを分離したり，アプリケーシ

† 芝浦工業大学 大学院理工学研究科 電気電子情報工学専攻
Division of Electrical Engineering and Computer Science,
Graduate School of Engineering and Science, Shibaura Institute of Technology

ョンが利用できる機能を制限したりするサンドボックスを実現している．そして，割り当てられた GID を基に機能の制限を行っているのが権限システムである．権限システムによりアプリケーションが利用できる機能は，ユーザに許可された権限を要求する機能のみに制限することが可能になる. また, 情報は機能を利用することで取得できるため，情報も GID によって制限されていることになる．

2. 権限システムの問題

権限システムには，次に示す問題が存在するため権限の確認だけでは，アプリケーションがマルウェアかどうか判断することは難しい．

A) 権限の粒度が粗く，一つの権限が許可する機能が多すぎるために，一部機能を許可するために拒否したい機能まで許可する必要がある．また，ユーザは権限が許可する機能のすべてをインストール時に確認することができない．
   - 例えば，電話の着信や通話の状態の取得は権限 READ_PHONE_STATE により制限されているが，この権限を許可することで他に，電話番号の取得や端末固有 ID の取得など多くの機能が使用可能になる．

B) アクセスが制限されるべき外部ストレージへの読込みについての権限が機能していないため，アプリケーションはユーザの許可なしに読込むことができる．
   - 外部ストレージへの書込みは権限 WRITE_EXTERNAL_STORAGE によって制限されているが，アプリケーションは読込みの権限 READ_EXTERNAL_STORAGE の許可を要求しなくても外部ストレージを読込むことができる．

C) 他のアプリケーションと連携する機能であるインテントを利用する事で，インターネット通信や，SMS 送信などの権限を要求しないアプリケーションでも，データを外部に送信できる．
   - 例えば，インターネット通信を行うための権限 INTERNET が許可されていない電話帳アプリケーションは，インテントを利用し Web ブラウザアプリケーションにコンタクトリストのデータを渡すことで，データを外部へ送信できる．

D) アプリケーションをインストールするためには要求される権限すべてを許可する必要があり，個別に許可・拒否することができない．

本稿の目的は，機能に関連するユーザ情報を明確にし，機能単位でコントロールすることで，権限システムが抱える問題を解決し，ユーザに許可された権限の範囲で悪事を働くマルウェアからユーザ情報を保護することである．

3. 関連研究

関連研究では，権限システムの問題を解決しマルウェアからユーザ情報を保護するために次のような提案がされている．

M.Ongtang ら[2]は，事前にインストールを許可するアプリケーションの情報（シグネチャやバージョン，要求する権限など）や，連携を許可するアプリケーションに要求するアプリケーションの情報，端末情報（バッテリーの状態や位置情報など）をセキュリティポリシーとして定義し，アプリケーションのインストール時と実行時にセキュリティポリシーと照らし合わせ検査を行う Saint（Secure Application INTeraction）というシステムを提案している．連携先のアプリケーションを制限できるため権限システムの問題 C)が起きないようにすることが可能である．しかし，システムを利用するに当たりアプリケーションに変更が必要になるという問題が存在する．

G.Russello ら[3]は，事前に定義したセキュリティポリシーを元にアプリケーションの動作やデータへのアクセスを制限する YAASE（Yet Another Android Security Extension）というシステムを提案している．データへのアクセス制限は，データにラベルを付加し，そのラベルのフィルタリングによって実現している．セキュリティポリシーには，アプリケーションが利用できるデータのラベルや，データを外部へ送信できる通信先などが定義されている．データがラベリングされているため，インテントによりデータが他のアプリケーションに渡されてもそのアプリケーションにデータの使用が許可されていなければ権限システムの問題 C)は発生しない．しかし，データにラベルを付加しフィルタリングを行うため，Android システムのパフォーマンスが低下するという問題が存在する．

Saint や YAASE では，事前にアプリケーションに許可する動作を定義し，それに基づいてアプリケーションを動作させるという静的な方法を取るが，静的な方法ではあらかじめ定義されたものにしか対応できず，アプリケーションを実行する度に変化するユーザが入力する SMS のメッセージや動的に生成される URL のような事前に定義できないものについては対応できないという問題が存在する．

これに対し川端ら[4]は，権限を要求する動作毎に承認を要求するダイアログを表示し，ユーザに承認の許可・拒否を求める機能や情報へのアクセス制御機構という動的にアプリケーションの動作を許可する方法を提案している．承認の結果は，承認を行ったアプリケーションのセキュリティポリシーとして保存され，次回以降はそのセキュリティポリシーを基に動作が実行・中断される．動作毎に承認を行うため，アプリケーションに許可する動作，権限を細かくコントロールでき，権限システムの問題 A)，D)を解決できる．しかし，権限を要求する動作の実行時に表示される

ダイアログには，動作を実行するアプリケーションの情報がアプリケーション名とアイコンしか表示されないため，同じアプリケーション名とアイコンを使用するアプリケーションが複数存在した場合，パッケージ名のようなアプリケーションを一意に特定するユニークな情報がないため，どのアプリケーションによる動作なのか判断できない．また，ユーザが確認できるのは実行される動作のみで，動作が使用するユーザ情報を確認することはできないという問題が存在する．他にも，すべての権限を要求する動作に対して，承認を行うため煩わしいという問題も存在する．

4. Android における脅威

4.1 脅威と資産の観点からの権限の分類

権限は複数の機能と関連を持つが，それを許可することがユーザにとって脅威となるかが，許可を判断する上での重要な情報である．機能は様々なユーザ情報を利用するため，資産となるユーザ情報やシステムの可用性に対する脅威の発生という観点から権限を分類した．その結果，権限は漏洩・改竄・課金・支配の 4 種類の脅威と，また単体では脅威には成りえないが脅威のある権限と共存することで危険になるユーザの情報である資産に分類できた．権限の分類は，権限名や権限が属する権限のグループ permissionGroup などから設定したルールに則り行う．なお，一つの権限に複数の脅威が存在する場合や，脅威には無関係な権限が存在する場合がある．

- 漏洩：外部へ資産を送信・公開する
  - ネットワークに接続する，または外部ストレージに書込む権限が該当する
  - INTERNET，WRITE_EXTERNAL_STORAGE など
- 改竄：資産を書換える
  - 権限名に WRITE，ADD，MODIFY，CHANGE，SET など生成･更新･削除を意味する単語が付き，資産に対して処理を行う権限が該当する
  - WRITE_CONTACTS，MANAGE_ACCOUNTS など
- 課金：料金が発生する
  - permissionGroup が COST_MONEY の権限，または NETWORK で 3G，LTE，WiMAX などを利用する権限が該当する
  - SEND_SMS，CALL_PHONE など
- 支配：ユーザの意思に関係なく端末の操作・設定の変更を行う
  - ハードウェアを利用する権限，システムやアプリケーションの自動操作を可能にする権限，ハードウェアやシステムの設定を変更する権限が該当する
  - CAMERA，RECORD_AUDIO など
- 資産：ユーザ情報を取得する
  - 権限名に ACCESS，GET，READ，RECEIVE など読取，取得を意味する単語が付き，ユーザ情報を取得する権限が該当する
  - READ_CONTACTS，GET_ACCOUNTS など

4.2 マルウェアによる脅威と機能・権限・資産の関係

電話帳アプリケーションを装ったマルウェアを例に，このマルウェアをインストールした場合に発生する脅威を拡張したミスユースケース図[5]で表すと図 1 のようになる．

このマルウェアには，SMS 送信，電話発信，コンタクトリストの取得といった機能が存在する．そして，これらの機能は氏名，電話番号，メールアドレスといった資産にアクセスすることが可能である．しかし，これらの機能を実行するために権限の許可が要求され，SMS 送信は SEND_SMS，電話発信は CALL_PHONE，コンタクトリストの取得は READ_CONTACTS がそれぞれ必要である．

権限は，脅威からユーザを保護するために存在するが，アプリケーションのインストールによって要求される権限はすべて許可されるため，図 1 のように SMS 送信を実行することによる漏洩や課金の脅威，電話発信を実行することによる課金の脅威が現れる．

脅威から資産を守るために，SMS 送信や電話発信に対して 5 節で提案する承認システムを実行する．これにより承認実行時に電話発信の発信先や，SMS 送信により送信される資産を確認でき，ユーザ情報が不正に外部へ送信されることを防ぐことが可能になる．

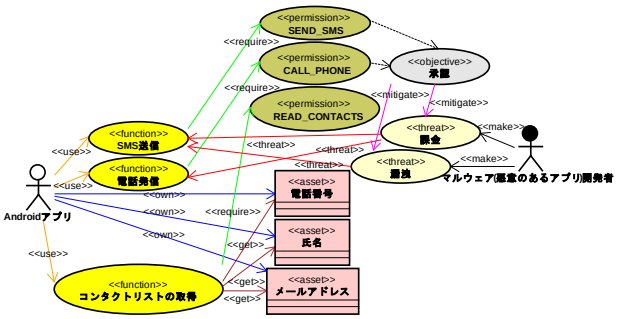

図 1　マルウェアによる脅威の分析

5. アプリケーションに対する動的なコントロールシステムの提案

本研究では，マルウェアからユーザ情報を保護するために，アプリケーションによって権限を要求する動作が実行される時，動作を実行させるかどうかの承認を動的にユーザに求め，承認の結果を基にアプリケーションの挙動を決定する動的アプリケーションコントロールシステムを提案する．承認要求時には，権限を要求する動作の説明やその動作がアクセスするユーザ情報，通信を行う場合の通信先などの情報をユーザが理解できる形で提示する．動作が使

用するユーザ情報が確認できることにより，実行される動作が不正かつ有害でないかを判断することが可能になる．また，動的に承認を実行することで，権限毎ではなく動作毎にアプリケーションの挙動を把握・コントロールすることが可能になる．

5.1 承認の煩わしさの解消

本提案システムでは権限を要求する動作毎に承認を求めるが，Android 4.1.2 では権限が 180 個存在するため，すべての権限について承認を行うことや，権限を要求する動作を実行する度に承認を行うことは煩わしくユーザビリティが低いという問題が存在する．この問題を解決するために，承認を行う権限を削減する方法と承認を行う回数を削減する方法で問題を解決する．

5.1.1 承認を必要とする権限の削減

権限にはリスクの強弱や署名を必要とするかなどの性質を示す protectionLevel が設定されている．マルウェアとして感染し得るユーザアプリケーション（ユーザがアプリケーションマーケットなどからインストール可能なアプリケーション）が利用できる権限は，protectionLevel が normal あるいは dangerous のものだけであるため，それらの権限を要求する動作のときのみ承認を行う．さらに 4.1 の脅威に該当する権限のときのみ承認を行う．これらにより承認を行う権限を 49 個まで削減することができた．なお，ユーザ情報にアクセスするための権限は，単体では脅威はないため，承認を実行しない．

5.1.2 権限承認の回数の削減

権限を要求する動作の実行時に，次のシステム情報から構成される動作についての「状態」を取得・保持する．

- 動作を実行するアプリケーションについての情報
  - アプリケーションのパッケージ名
  - アプリケーションのバージョンコード
  - 権限を要求する動作を呼び出しているアプリケーション内のメソッド名
  - メソッドが定義されているクラス名
  - クラスが定義されている Java ファイル名
  - ファイル内でメソッドが定義されている行番号
- 権限を要求する動作についての情報
  - 権限を要求する動作名
  - 要求されている権限名
  - メソッドに渡される引数の値（動作に使用されるユーザ情報や通信を行う場合の通信先など）

これらの情報から，アプリケーションのソースコード中のどこから動作が実行され，その動作を実現するためのメソッドに渡される引数などがわかるため，権限を要求する動作やその動作を実行するアプリケーションを一意に特定できる．よって，再度同じ「状態」で権限を要求する動作が実行され，その動作が不正かつ有害でないとユーザが判断した場合，承認を実行せず以前ユーザが行った承認の結果を基に自動的に動作の実行・中断を行うことができる．これにより承認の回数を削減する．なお，次回以降承認を行わず自動で実行するかは承認実行時にユーザに尋ねる．

ユーザの選択，承認スキップ（承認を実行せず以前の承認結果を基に動作を実行・中断すること）の可否による結果は表 1 のようになる．

表 1　ユーザの選択，承認スキップの可否による結果

ユーザの選択＼承認スキップ	有効	無効
許可	動作実行	承認要求
拒否	動作中断	
選択なし	動作中断	

5.2 承認要求通知

承認を要求する際，ダイアログを使用し通知を行う．ダイアログには，次のような情報を表示する．これらの情報から，権限を要求する動作を実行するアプリケーションが一意に特定でき，また実行される動作，動作が使用するユーザ情報や通信を行う場合の通信先が分かる．よって，これを基に不正かつ有害な動作でないかを判断する．

- どのアプリケーションが権限を要求する動作を実行しようとしているのかを特定するための情報
  - アプリケーション名
  - アプリケーションのアイコン
  - アプリケーションを一意に特定するためのパッケージ名
- 権限を要求する動作についての情報
  - 権限を要求している動作の説明
  - 要求されている権限名
  - 動作に使用されるユーザ情報の説明
  - 通信を行う場合はその通信先

承認が要求されるのは，常にユーザが端末に触れている時（Activity 起動時）とは限らず，Service や Broadcast Receiver によってバックグラウンドで実行される時にも起こり得る．そのためユーザの知らぬ間に意図しない動作が実行されることを防ぐため，承認が要求されてから一定時間，ユーザによる操作がない場合，自動的に承認は拒否される．そして，権限を要求する動作が実行されたことや自動で承認が拒否されたことをステータスバーに通知する．これによりユーザは知らぬ間に動作が実行されたことを知ることができる．

5.3 実装

承認の煩わしさの解消と承認実行時に動作に使用されるユーザ情報を提示するために，Android のアプリケーション開発においてメインで利用される Java 言語向けの API（Application Programming Interface）を提供するアプリケーションフレームワーク層を拡張し，本提案システムを実現する．そのため，C や C++といったプログラミング言語で

開発されたアプリケーションには対応することができない．また，提案するシステムは実用性を考慮し，既存のアプリケーションに変更を要求することなく目的を達成できるものとする．

図 2 に本提案システムの流れを示す．

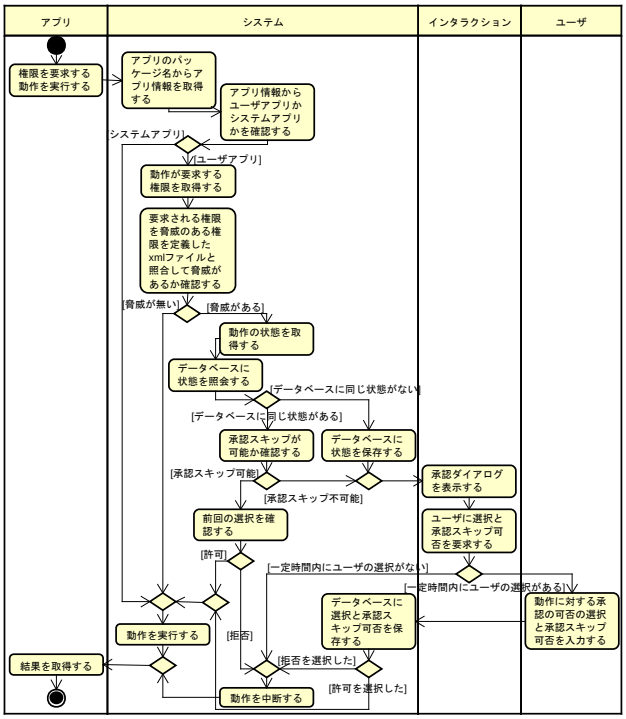

図 2　システムの流れ

5.3.1 動作の捕捉

動的にユーザに承認を求めるために漏れなく実行される動作を捕捉する必要がある．Android において，動作は次の 3 パターンから実行されため，動的に承認を行うためにこれらのパターンをフックする．

- インテント
  - 電話発信は権限 CALL_PHONE を要求し，他のアプリケーションと連携する機能であるインテントにアクション Intent.ACTION_CALL と電話番号を指定することで実行できる
- コンテントプロバイダ
  - コンタクトリストの取得は権限 READ_CONTACTS を要求し，アプリケーションが管理しているデータをほかのアプリケーションにも提供する機能であるコンテントプロバイダに URI（Uniform Resource Identifier）の ContactsContract.Contacts.CONTENT_URI を指定することで取得できる
- メソッド
  - SMS 送信は権限 SEND_SMS を要求しメソッド SmsManager#sendTextMessage()に送信するメッセージと送信先を指定することで実行できる

承認実行後，承認結果が許可なら動作を実行させ，拒否なら動作の実行を中断させる．動作の実行を中断する際，動作がデータの取得など戻り値を取得するものであれば，null や 0 など動作を正常に実行した場合，得ることのない値を返す．API によっては，異常時に取得できる戻り値が定義されているため，それを利用する．

5.3.2 情報の取得

承認の煩わしさの解消と承認実行時に動作に使用されるユーザ情報の提示に必要になる情報は次のように取得する．

- アプリケーション名，パッケージ名，バージョンコード，アプリケーションのアイコン
  - Android の API から取得する
- メソッド名，クラス名，Java ファイル名，行番号
  - アプリケーションのスタックトレース（プログラムの実行過程を記録した情報）から取得する
- 動作名や動作の説明，権限名，メソッドの引数（動作に使用されるユーザ情報や通信先）
  - 動作を実行するメソッドから取得できるようにメソッドを拡張し，java.lang.String#valueOf()メソッドを用いて文字列化し取得する
  - 動作名や動作の説明は xml ファイルに定義されている

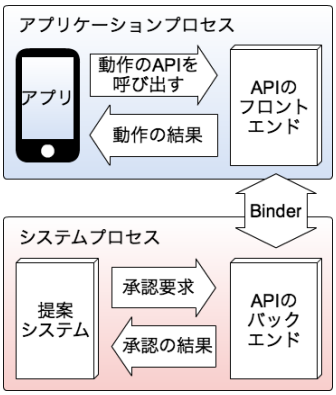

図 3　システム呼出し手順

Android のサンドボックスによって，システムとアプリケーション間の干渉を防ぐためにシステムのプロセスとアプリケーションのプロセスは分離されている．プロセス間の通信（IPC : Inter Process Communication）は，RPC（Remote Procedure Call）のフレームワークである Binder を利用し実行される．

API はアプリケーションプロセスで呼び出される部分（フロントエンド）と RPC を用いてシステムプロセスで実行される部分（バックエンド）に分かれる．本提案システムは Android のシステムプロセス上で実行されるが，アプリケーションのスタックトレースを取得するためにアプリケーションプロセス上で実行される API の呼び出し部分で

スタックトレースを取得する．よってシステムの呼び出し方は図 3 のようになる．

動作名や動作の説明は，脅威のある権限とその権限を要求する動作について定義した xml ファイルから取得する．それらは xml ファイルに定義され，端末起動時にシステムによって読込まれる．xml ファイルは次のように記述する．

```
<!-- 脅威のカテゴリの定義（カテゴリ，説明） -->
<threat-group name="android.threat-group.LEAK"
              label="@string/threat_group_label_leak"
        description="@string/threat_group_descr_leak" />

<!-- 脅威のある権限の定義（権限名，有する脅威） -->
<!-- 脅威が複数ある場合は「|」で区切る -->
<threat-permission name="android.permission.SEND_SMS"
              threatGroup="leak|billing|control" />

<!-- 動作の定義（動作名，要求する権限，説明） -->
<threat-action name=
              "android.telephony.SmsManager#sendTextMessage"
        permission="android.permission.SEND_SMS"
              descr="@string/threat_action_send_text_sms" />
```

6. 動作検証

本提案システムにより，権限システムや関連研究が抱える問題に対して有効であるか図 1 のマルウェアを例に検証を行う．

- 電話番号 090XXXXXXXX へメッセージ「Hello!」をメソッドにより実行される SMS 送信を実行したとき，次の情報を表示した図 4 のダイアログが表示された．
  - アプリケーションの情報（アプリケーション名やパッケージ名，アイコン）
  - 動作の説明：「SMS 送信を行います」
  - SMS 送信先電話番号：「090XXXXXXXX」
  - 送信するメッセージ：「Hello!」
- 電話番号 080YYYYYYYY へインテントにより実行される電話発信を実行したとき，次のような情報がダイアログに表示された．
  - アプリケーションの情報
  - 動作の説明：「電話発信を行います」
  - 発信先電話番号：「080YYYYYYYY」

このマルウェアは起動後一定時間後に Broadcast Receiver から自動で取得したコンタクトリストをユーザに無断で不正に SMS 送信するが，そのような場合でも上記のような情報が表示されたダイアログが表示された．

承認を許可することで動作の実行，拒否することで動作が中断されたことも確認できた．また，承認スキップにより承認回数を減らすこともできた．

図 4　承認要求ダイアログ

7. まとめ及び今後の方針

既存のアプリケーションに変更を必要とせず動作するように，Android のアプリケーションフレームワークを拡張し本提案システムを実装した．承認によって権限を要求する動作の実行結果が変わるため整合性を保つため動作毎に拡張する必要があり，現時点では一部の権限にしか対応していないが，対応している権限を要求する動作については，承認画面に表示される情報から不正かつ有害な動作でないかの判断をすることができる．また，動的に動作毎に承認を行い，動作が使用するユーザ情報を提示することでアプリケーションが実行する動作やアクセスするユーザ情報に対する制御を動作やユーザ情報毎に行うことが可能になり，権限システムの問題や関連研究が抱える問題を解決することができた．

今後は，表示される情報が十分であるか，また情報からマルウェアをマルウェアであると判断できるかの検証，更なる権限承認の煩わしさの解消の模索，対応していない権限への対応，全権限への対応後に懸念されるパフォーマンスの低下の改善を課題とし，研究を進める予定である．

参考文献

[1] Android Developers, http://developer.android.com/
[2] M.Ongtang, S.McLaughlin, W.Enck and P.McDaniel: Semantically Rich Application-Centric Security in Android, ACSAC, pp.340-349 (2009).
[3] G.Russello, B.Crispo, E.Fernandesa and Y.Zhauniarovich: YAASE: Yet Another Android Security Extension, PASSAT and SocialCom, pp.1033-1040 (2011).
[4] 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, Computer Security Symposium, pp.161-166 (2011).
[5] G.Sindre and A.L.Opdahl: Eliciting security requirements with misuse cases, Requirements Enginnering Journal, 10(1), pp.34-44 (2005).