Vol.2013-CSEC-62 No.23
Vol.2013-SPT-6 No.23
2013/7/18

Androidにおけるユーザの意図しない情報の漏洩を防止するパーミッション動的制御

渡邊 華奈子[1,a)]　大月 勇人[1]　瀧本 栄二[2]　齋藤 彰一[3]　毛利 公一[2]

概要：Android は，パーミッションと呼ばれる権限によって，アプリケーションによる端末内の情報へのアクセスの可否を判断している．ユーザは，アプリケーションのインストール時に，アプリケーションへ与えるパーミッションを確認することができる．しかし，実際には，要求されたパーミッションを全て付与しなければアプリケーションを利用することができないため，ユーザは拒否しづらい．また，一度パーミッションを与えると，アプリケーションが情報へのアクセスする際にユーザには通知はされない．よって，悪意あるアプリケーションは実行時にユーザに無断で情報を漏洩させることができる．以上から，アプリケーションが情報の取得や漏洩に関わる動作を行うときにユーザに通知する機構を実現した．この機構により，ユーザはアプリケーションによる情報の取得や漏洩に関わる動作の実行可否を選択することが可能である．これにより，アプリケーションがユーザに無断で情報を漏洩させることを防ぐ．

Dynamic Permission Control to Prevent Unexpected Information Leakage on Android

Kanako Watanabe[1,a)]　Yuto Otsuki[1]　Eiji Takimoto[2]　Shoichi Saito[3]　Koichi Mouri[2]

Abstract: Android controls accesses to privacy information by "permission". A user can check what permissions will be granted to an application on its installation. However, the user must grant all the permissions requested by the application to install. Thus, it is difficult for the user to refuse granting the permissions. Furthermore, android does not notify the user that the application is acquiring privacy information at runtime. Therefore, malicious applications can leak privacy information without a notification to the user. This paper describes implementation of dynamic permission controller that notifies a user that an application is acquiring privacy information at runtime. This function achieves that the user can select the application's behavior, such as acquiring privacy information or not. As a result, privacy information leakage without user's intention will be preserved.

1. はじめに

Android[1] は，ユーザがアプリケーションをインストールして利用できるスマートフォン・タブレット端末向け OS である．アプリケーション開発者は，Google 社が公開しているアプリケーション開発ツールを入手することで，アプリケーションを作成し，公開することが可能である．このため，ユーザの個人情報を漏洩させるアプリケーションが作成・公開されるケースも多く，被害が発生している．

Android において，アプリケーションが端末内に保存されているアドレス帳や電話番号などの個人情報を取得し

[1] 立命館大学大学院情報理工学研究科
Graduate School of Information Science and Engineering, Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577 JAPAN
[2] 立命館大学情報理工学部
College of Information Science and Engineering, Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577 JAPAN
[3] 名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 JAPAN
[a)] knkedge@asl.cs.ritsumei.ac.jp

たり，ネットワークを利用したりするためには，パーミッションと呼ばれる権限がそのアプリケーションに付与されている必要がある．アプリケーションのインストール時に，ユーザはアプリケーション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．しかし，ユーザは，それらすべてを付与することに同意しなければアプリケーションをインストールして利用することができない．また，一度端末にインストールされたアプリケーションは，付与されたパーミッションに対応する情報や機能を制限なく常時利用することができる．さらに，それらの動作がユーザへ通知されることはない．すなわち，アプリケーションは，ユーザに気づかれることなく個人情報を漏洩させることが可能である．

以上から，アプリケーションによる情報漏洩に関わる動作について，実行時にユーザに同意を得ることで問題を解決する．具体的には，アプリケーションが情報漏洩に関する API（Application Programming Interface）を呼び出したときに，ユーザに対して通知するとともに実行の可否を選択可能とするパーミッション動的制御機構を提案する．また，API 呼出しごとの確認はユーザにとって負担となるため，利便性を考慮し，アプリケーションごとに API 実行の可否を事前に設定できる通知設定機能も実装した．これによって，インストールされたアプリケーションがユーザに気づかれることなく情報を漏洩させることを防ぐことができる．

以下，本論文では，2 章で Android のアクセス制御について，3 章でパーミッション動的制御機構について述べる．4 章でパーミッション動的制御機構の実装について述べ，5 章で動作検証の結果について述べる．6 章で関連研究について述べ，7 章でまとめる．

2. Android のパーミッションとその問題点

2.1 サンドボックス

Android は，アプリケーションが他のアプリケーションのデータへアクセスすることを制限している．すべてのアプリケーションには，アプリケーションごとに Linux のユーザ ID が割り当てられている．アプリケーションは，割り当てられた Linux のユーザ権限のプロセスとして動作する．また，おのおののアプリケーションのデータが格納されるディレクトリは，当該アプリケーションのユーザ ID を所有者とし，所有者のみがアクセス可能となっている．以上のように，Android では，アプリケーションが他のアプリケーションのデータを直接アクセスすることを禁止している．

2.2 パーミッション

2.1 節で述べたとおり，アプリケーションは他のアプリケーションのデータに直接アクセスすることはできない．

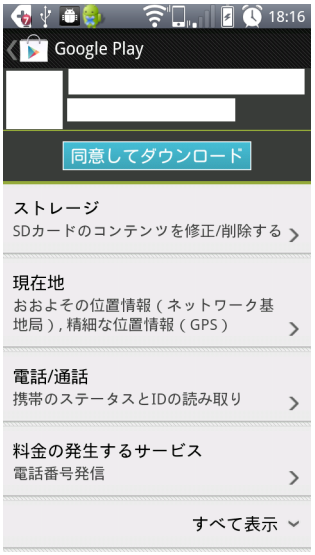

図 1 インストール時に表示されるパーミッション確認画面

Android では，ネットワーク接続やアドレス帳のような特定の機能や情報は，特定のアプリケーションがそれぞれ対応した API を介して提供している．アプリケーションがこれらの API を呼び出すためには，API ごとに対応したパーミッションがそのアプリケーションに付与されている必要がある．

アプリケーションに付与されるパーミッションの一覧は，アプリケーションのインストール時に，図 1 のような確認画面として表示される．なお，ここで表示されるパーミッションは，アプリケーション開発者が AndroidManifest と呼ぶファイルに記述したものである．ユーザは，表示されたパーミッションをアプリケーションに付与することに同意することでアプリケーションをインストールして利用することができる．

アプリケーションが実行されて API を呼び出すと，対応したパーミッションが付与されているか確認が行われる．アプリケーション A がアドレス帳のデータを取得する場合の動作を図 2 に示す．アプリケーション A がアドレス帳 API を呼び出すと，パーミッションチェック機構が，アドレス帳のデータの取得に必要な READ_CONTACTS パーミッションが付与されているか確認を行う．アプリケーション A に READ_CONTACTS パーミッションが付与されていれば，アドレス帳管理アプリケーションが，インタフェースを介してアプリケーション A にアドレス帳のデータを提供する．

2.3 問題点

Android では，2.2 節で述べたような仕組みを用いて，アプリケーションによる個人情報へのアクセスを可能にしている．しかし，これには次の 2 つの問題が存在する．1 つ目は，ユーザは，アプリケーションを利用するために，アプリケーションに必要なパーミッションを全て付与するこ

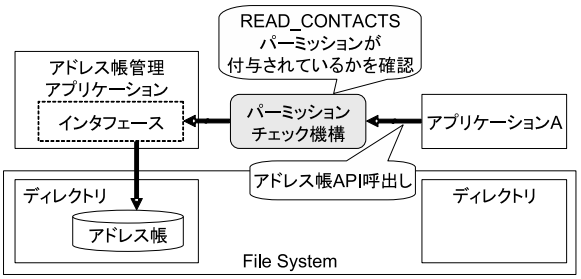

図 2 アプリケーションによるアドレス帳の読出し

とに同意しなければならないことである．また，ユーザは，アプリケーションによる情報の取得の可否をインストール後に変更することができない．2 つ目は，アプリケーションが，パーミッションを必要とする API を呼び出したことをユーザが知ることができないという点である．そのため，Android では，インストールされたアプリケーションが個人情報にアクセスし，漏洩させたことをユーザが知ることができない．

これらの 2 つの問題点から，一度インストールされたアプリケーションは，常にユーザに気づかれることなく個人情報を取得・漏洩させることが可能となっている．

3. パーミッション動的制御機構

3.1 概要

本論文では，2.3 節で述べた 2 つの問題点について，インストール後にユーザが API 呼出しの可否を選択できるようにすることで解決する．Android では，アプリケーションがパーミッションを必要とする API を呼び出すと，そのパーミッションがアプリケーションに付与されているかを確認する．そこで，Android のパーミッションチェック機構内にパーミッション動的制御機構（以下，DPC と呼ぶ）を構築する．DPC は，アプリケーションが情報の取得・漏洩に関わる動作を実行するときに，ユーザによる動作の実行可否の選択を可能にする．

DPC を追加したパーミッションチェック機構の動作を図 3 に示す．DPC は，アプリケーションが情報の取得・漏洩に関わる API を呼び出したとき（図 3(1)），画面にダイアログを表示し実行の可否をユーザに尋ねる（図 3(2)）．ユーザは，それに対して可否を選択する（図 3(3)）．ユーザが許可を選択すると API は実行され，ユーザが拒否を選択すると API は実行されない．

さらに，DPC は，アプリケーションによる情報の取得や外部への送信などユーザの意志が事前に決定しているケースについて，実行の可否を毎回尋ねることがないように設定できる通知設定機能を提供する．ユーザは，通知設定用のアプリケーション（以下，通知設定 AP と呼ぶ）を用いて通知設定機能の設定ができる（図 3(4)）．設定可能な内容は，「常時許可」，「常時拒否」，「毎回確認」のいずれかである．これを既定値設定と呼ぶ．DPC は，ダイアログを

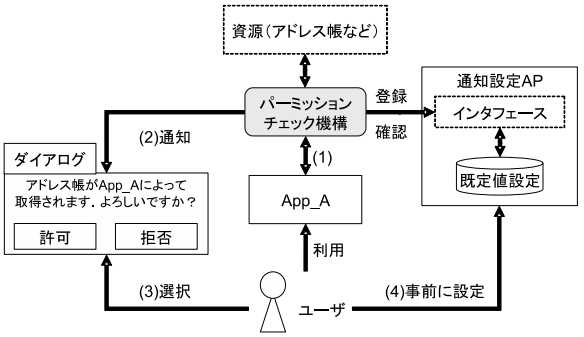

図 3 パーミッション動的制御機構を追加したときの動作

表示する前に通知設定 AP に保存されている既定値設定を確認し，毎回確認のときのみダイアログを表示してユーザに可否を問う．常時許可・常時拒否の場合はユーザに問うことなく API の成否を決定する．

3.2 個人情報に関わるパーミッション

総務省は，スマートフォンプライバシイニシアティブ [2] において，アプリケーション開発者や広告業者に対して適切な取扱いをすべき情報として，電話帳データや位置情報などを利用者情報として挙げている．本論文では，この利用者情報を保護すべき個人情報とする．これらの情報を取得するためには，Android 2.3.4 では，表 1 に示すパーミッションが必要である．DPC は，これらのパーミッションを制御対象とする．

また，表 1 のパーミッションが付与されることによって取得された個人情報がネットワーク上へ送信されるためには，別途表 2 に示すパーミッションが必要である．これらのパーミッションは，アプリケーションが情報をインターネットや SMS を利用するために必要である．よって，これらのパーミッションを必要とする API 呼出しが行われた場合も，ユーザに対してダイアログを表示し，API の実行可否を尋ねる．

4. 実装

3 章で述べた DPC を，Android 2.3.4 のパーミッションチェック機構に実装した．なお，現在は，表 1，表 2 に示したパーミッションのうち，READ_PHONE_STATE，READ_CONTACTS，ACCESS_FINE_LOCATION，ACCESS_COARSE_LOCATION，INTERNET の 5 つのパーミッションの処理の実装が完了している．DPC の動作は，(1) 既定値設定の確認，(2) ダイアログによるユーザへの実行可否の結果確認，(3) パーミッションチェックへの反映から成る．(1) は通知設定 AP を作成することで実現し，(2)，(3) はパーミッションチェックを行う ActivityManagerService を拡張することによって実現した．

表 1 利用者情報とパーミッションの対応

パーミッション名	取得する情報／利用する機能	対応する利用者情報
READ_PHONE_STATE	端末の固有情報（端末識別子，電話番号など）	契約者・端末固有 ID
PROCESS_OUTGOING_CALLS	通話発信の監視	通話履歴
READ_CONTACTS	アドレス帳に登録されている情報	電話帳データ，通話履歴
READ_LOGS	端末のシステムログの読取り	電話帳データなど
GET_ACCOUNTS	端末内のアカウント	契約者固有 ID
CAMERA	カメラ撮影	映像・写真
ACCESS_FINE_LOCATION	GPS などから取得する位置情報	位置情報
ACCESS_COARSE_LOCATION	Wifi や基地局などから取得する位置情報	位置情報
READ_HISTORY_BOOKMARKS	ブラウザのお気に入り，閲覧履歴	ネット閲覧履歴

表 2 ネットワークへの情報送信に利用されるパーミッション

パーミッション名	機能
INTERNET	ネットワークへの接続
SEND_SMS	SMS の送信

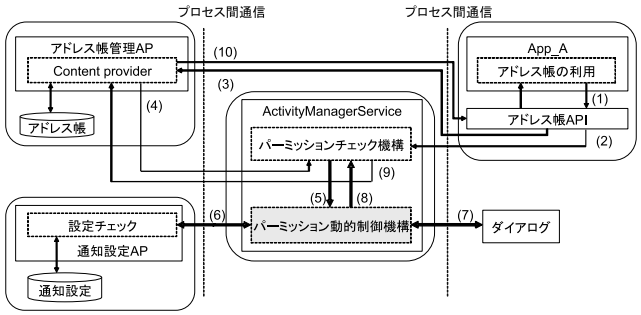

図 4 パーミッション動的制御機構の動作

4.1 動作

DPC が動作するタイミングと，動作の流れを図 4 に示す．図 4 は，アドレス帳のデータを読み出すために必要な READ_CONTACTS パーミッションを付与された App_A が，アドレス帳のデータを提供するアドレス帳管理 AP に対してアドレス帳のデータを要求する場合の動作の例を示している．図 4 の (5)–(8) が DPC の動作である．

(1) App_A が，アドレス帳のデータを読み出すためにアドレス帳 API を呼び出す．

(2) アドレス帳 API は，ActivityManagerService にパーミッションチェックを依頼する．ActivityManagerService は，アドレス帳管理 AP へのアクセスに必要な READ_CONTACTS パーミッションが App_A に付与されているかの確認を行う．その後，パーミッションチェックの結果をアドレス帳 API に返す．

(3) App_A に READ_CONTACTS パーミッションが付与されている場合は，アドレス帳 API がアドレス帳管理 AP に対してアドレス帳のデータを要求する．

(4) アドレス帳管理 AP のインタフェースである Content provider は，ActivityManagerService にパーミッションチェックを依頼する．ActivityManagerService は，App_A に READ_CONTACTS パーミッションが付与されているかの確認を行う．

(5) 既存のインストール時に設定されたパーミッションのチェックにおいて App_A に READ_CONTACTS パーミッションが付与されていることが確認された場合，DPC に制御を移す．

(6) DPC は，通知設定 AP と同期通信を行い，ユーザの既定値設定を確認する．App_A による READ_CONTACTS パーミッションの利用時に常時許可の設定になっていれば，DPC はパーミッションチェックに許可を返す．

(7) App_A による READ_CONTACTS パーミッションの既定値設定が毎回確認になっている場合，DPC は，ユーザに対してダイアログを表示し，アドレス帳の読出しの実行の可否を尋ねる．そして，ダイアログは，ユーザが可否を選択した結果を DPC へ送信する．

(8) DPC からパーミッションチェック機構にユーザが選択した結果を返す．ユーザが処理の実行を拒否した場合は，パーミッションチェック機構の結果を許可から拒否に書き換える．

(9) パーミッションチェックの結果をアドレス帳管理 AP の Content provider に返す．

(10) パーミッションチェックの結果が許可ならば，アドレス帳管理 AP が App_A にアドレス帳のデータを返す．パーミッションチェックの結果が拒否ならば，App_A によるアドレス帳データの要求を拒否する．

4.2 パーミッションチェックのフック

本節では，アドレス帳管理 AP がアドレス帳のデータを提供するインタフェースである Content provider を例として，図 4(5) に示したパーミッションチェックをフックする手法について述べる．App_A のアドレス帳データ要求によるパーミッションチェックの呼出しの流れを図 5 に示す．アプリケーションによるアドレス帳のデータ要求

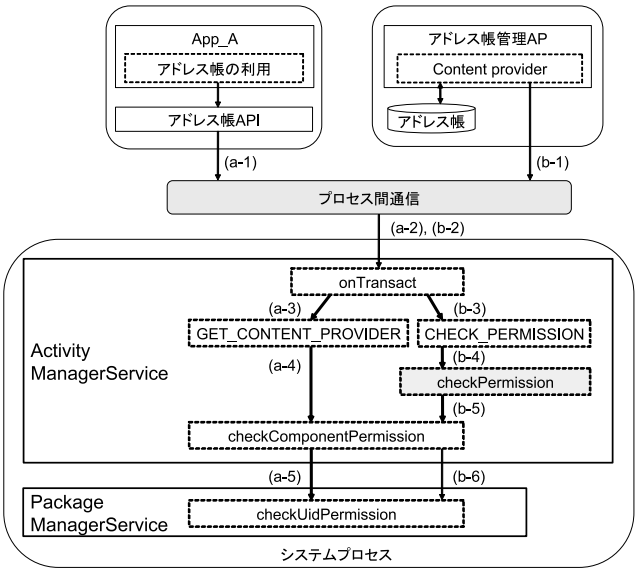

図 5 アドレス帳データ要求時に呼び出されるパーミッションチェック

では，パーミッションチェックが 2 回行われる．1 回目のパーミッションチェックは，App_A が Content provider へのアクセスを行う前にアドレス帳 API が依頼する（図 5(a-1)–(a-5)）．2 回目のパーミッションチェックは，アドレス帳管理 AP がアドレス帳のデータを返す前に依頼する（図 5(b-1)–(b-6)）．

アドレス帳 API に依頼される 1 回目のパーミッションチェックの動作を以下に示す．

(a-1) アドレス帳 API は，アドレス帳管理 AP の Content provider へアクセスするために，GET_CONTENT_PROVIDER メッセージを ActivityManagerService に送信する．

(a-2) プロセス間通信で ActivityManagerService の onTransact メソッドにメッセージが送信される．

(a-3) ActivityManagerService の onTransact メソッドが GET_CONTENT_PROVIDER メッセージに対する処理を呼び出す．

(a-4) GET_CONTENT_PROVIDER メッセージに対する処理から checkComponentPermission メソッドが呼び出される．checkComponentPermission メソッドは，アドレス帳管理 AP が他のアプリケーションからのアクセスを許可しているかチェックを行う．

(a-5) PackageManagerService の checkUidPermission メソッドが呼び出される．PackageManagerService はアプリケーションに付与されたパーミッションのデータを管理している．checkUidPermission メソッドでは，App_A に割り当てられた Linux ユーザ ID からアプリケーションにパーミッションが付与されているかチェックする．

アドレス帳管理 AP に依頼される 2 回目のパーミッションチェックの動作を以下に示す．

(b-1) アドレス帳管理 AP は，App_A に READ_CONTACTS パーミッションが付与されているかパーミッションチェックを行うために CHECK_PERMISSION メッセージを ActivityManagerService に送信する．

(b-2) プロセス間通信で ActivityManagerService の onTransact メソッドにメッセージが送信される．

(b-3) メッセージを受け取った ActivityManagerService の onTransact メソッドは，CHECK_PERMISSION メッセージに対する処理を呼び出す．

(b-4) CHECK_PERMISSION メッセージに対する処理から，checkPermission メソッドが呼び出される．

(b-5) checkPermission メソッドは，checkComponentPermission メソッドを呼び出す．

(b-6) checkComponentPermission メソッドは，PackageManagerService の checkUidPermission メソッドを呼び出す．

1 回目のパーミッションチェックは App_A が Content provider へのアクセスができるかどうかを確かめるものである．実際にアドレス帳のデータが返されるのは 2 回目のパーミッションチェックが行われた後である．そのため，DPC は 2 回目に行われるパーミッションチェックをフックする．ここで，1 回目のパーミッションチェックでは，checkPermission メソッドは呼び出されないが，2 回目のパーミッションチェックでは checkPermission メソッドが呼び出される．したがって，DPC では，実際に個人情報が取得される際に実行される checkPermission メソッドをフックするよう実装した．なお，アドレス帳 API 以外の API 呼出しにおいても，同様の場所でフックすることができる．

4.3 ダイアログの表示

Android が提供する画面操作の API は，シングルスレッドモデルである．パーミッションチェック機構では，パーミッションチェックが動作するスレッドと画面操作を行うスレッド（以下，GUI スレッドと呼ぶ）が異なる．したがって，ダイアログを表示するためには，パーミッションチェックスレッドから GUI スレッドに対してダイアログの表示を依頼する必要がある．この処理を，Android で提供されているスレッド間通信を用いて実装した．

図 6 に，アドレス帳管理 AP からパーミッションチェックを依頼されたときの動作を示す．DPC は，ダイアログの表示とユーザから受け取った実行可否のパーミッション

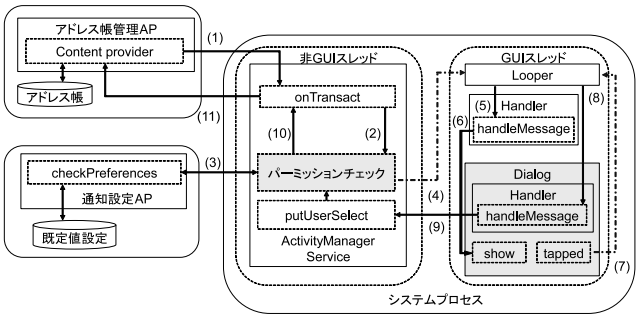

図 6 パーミッション動的制御機構の内部動作

チェックへの反映を以下の流れで行う．なお，図 4 では，(7) に該当する．

(1) アドレス帳管理 AP からパーミッションチェックが依頼されると，プロセス間通信のメッセージを受け取る onTransact メソッドが呼び出される．

(2) パーミッションチェックの依頼を受け取った onTransact メソッドからパーミッションチェックの処理が呼び出される．

(3) 既存のパーミッションチェックでパーミッションが付与されていることが確認された場合，DPC に制御を移す．通知設定 AP と同期通信を行い，毎回確認の設定であるか確認する．

(4) 設定が毎回確認であれば，ダイアログの表示を依頼するメッセージを GUI スレッドの Looper に送信する．Looper は，スレッド間で通信を行うときのメッセージの受け口である．

(5) ダイアログの表示を依頼するメッセージを受け取った Looper が，メッセージを処理する Handler を呼び出す．

(6) Handler が呼び出されるとユーザにパーミッションの利用を通知し，アドレス帳のデータの読出しの可否を尋ねるダイアログを表示する．

(7) ユーザが可否を選択すると，ダイアログはその結果をメッセージとして Looper に送信する．また，メッセージを処理する Handler を Dialog の Handler に指定する．

(8) ダイアログの選択結果のメッセージを受け取った Looper が，Dialog の Handler を呼び出す．

(9) Looper から呼び出された Dialog の Handler は，ActivityManagerService の putUserSelect メソッドを呼び出し，DPC にユーザが選択した結果を通知する．

(10) パーミッションチェックは，ユーザが選択した結果を反映したパーミッションチェックの結果を返す．

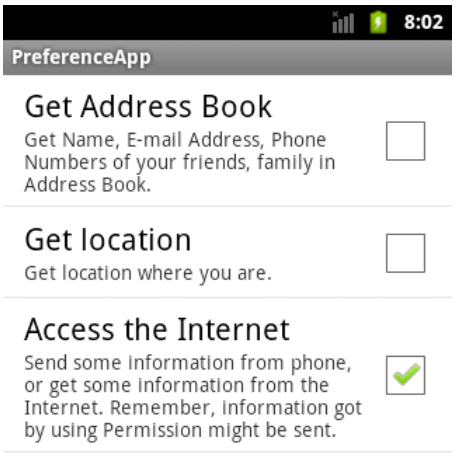

図 7 ユーザによる既定値設定画面

(11) onTransact メソッドからアドレス帳管理 AP にパーミッションチェックの結果を返す．

4.4 ユーザによる通知設定

通知設定 AP のプロトタイプとして，既定値を常時許可に設定するアプリケーションを作成した．通知設定 AP は，すべてのアプリケーションに対する既定値と個別のアプリケーションごとの既定値を保持する．また，DPC とユーザに対してそれぞれインタフェースを提供する．

ユーザに対するインタフェースを，図 7 に示す．図 7 は，すべてのアプリケーションに対する既定値を設定する画面である．ユーザは，この画面で既定値を常時許可に設定する API にチェックを入れる．アプリケーションごとの既定値についても，同様の設定画面を提供する．

DPC に対するインタフェースは，システムプロセスからのみアクセスすることができる．DPC は，呼び出される API と API を呼び出したアプリケーションを当該インタフェースに渡す．通知設定 AP は，そのアプリケーションと API に対して，ユーザの設定した既定値を確認し，DPC に返す．

5. 動作検証

5.1 検証内容

DPC の動作を確認するために，Android Emulator[3] を用いて検証を行った．DPC を実装した Android 上で Google Play[4] で公開されているアプリケーションを実行し，パーミッションが必要な動作が実行されるときの DPC の動作を確認した．動作させるアプリケーションには，広告を表示する Noogra Nuts[5] というゲームアプリケーションを用いた．アプリケーションを動作させたときに広告が記憶されている可能性を考慮し，アプリケーションの動作の確認を行うたびにゲームアプリケーションのデータの消去を行った．Noogra Nuts が利用するパーミッションは，以下の 4 つである．DPC では，INTERNET パーミッショ

図 8 アプリケーションの動作を通知するダイアログ

ンと READ_PHONE_STATE パーミッションを制御対象としている．

INTERNET
インターネットへのアクセスを許可する．

READ_PHONE_STATE
端末の識別子や電話番号，通話中の通話相手の電話番号の読み取りを許可する．

WRITE_EXTERNAL_STORAGE
SD カードへの書込みをアプリケーションに許可する．

WAKE_LOCK
端末のスリープを無効化する．

5.2 検証結果

このアプリケーションを実行することで，以下の 3 つの機能が実現されていることを確認した．

- パーミッションが必要とされるアプリケーションの動作の実行時に通知を行った．
- パーミッションが必要とされるアプリケーションの動作の実行可否を選択できた．
- 通知設定 AP から実行を常時許可する API を設定できた．

アプリケーションを起動すると，図 8 のパーミッションの使用を通知する画面が 3 回表示された．READ_PHONE_STATE パーミッションの利用時に通知を行う設定でアプリケーションを起動すると，図 8 と同様に，READ_PHONE_STATE パーミッションが必要な動作の実行がユーザに通知された．したがって，パーミッションが必要な動作がユーザに通知されることを確認した．

パーミッションが必要な動作の実行可否をユーザが選択できるかの確認を行った．表示された図 8 のダイアログに対して 3 回とも拒否すると広告は表示されず，3 回とも許可すると広告が表示された．このアプリケーションは，広告の画像をダウンロードするために INTERNET パーミッションを利用している．ダイアログで 3 回とも拒否を選択すると広告が表示されなかったことから，INTERNET パーミッションが必要な動作の実行をユーザが拒否することが可能になったことを確認した．

通知設定 AP の動作を確認するため，通知設定 AP で既定値設定を常時許可にした状態でのアプリケーションの動作も確認した．INTERNET パーミッション，READ_PHONE_STATE パーミッションともに既定値設定を常時許可で起動すると，ダイアログは表示されず，広告が表示された．INTERNET パーミッションのみ常時許可にした場合は，READ_PHONE_STATE パーミッションが必要な動作を通知するダイアログ画面が表示された．また，READ_PHONE_STATE パーミッションのみ常時許可にしてアプリケーションを起動すると，INTERNET パーミッションが必要な動作を通知するダイアログ画面が表示された．これらのことから，通知設定 AP によって一部のパーミッションが必要な動作のみを常時許可する設定が実現されていることを確認した．

6. 関連研究

アプリケーションが個人情報の取得やネットワークへ送信する API を呼び出したときにユーザに承認を得る研究がある．上松ら [6] は，アプリケーションが API から個人情報を取得するときに，個人情報ではなく個人情報を指すポインタを渡し，画面などに出力する際にポインタから個人情報を復元するシステムを提案している．このシステムでは，アプリケーションによる漏洩は防ぐことができるが，アプリケーションが個人情報を加工して利用することような場合には適用できない．また，加藤ら [7] は，ユーザがアプリケーションによるユーザ情報へのアクセスや Android 機能の利用を細かくコントロールすることを目的とした動的承認システムを提案している．このシステムは，利用された API の情報から，取得される情報や通信先などのユーザがアプリケーションの動作をマルウェアによるものかを判断する助けになる情報を画面に表示する．

上記の 2 つのシステムは，API 呼出しをフックすることで実現されている．API をフックすることによって，パーミッションよりも細かい単位で情報の取得を制御することが可能であるが，API それぞれにフックを実装する必要がある．また，Android NDK[8] などを利用したネイティブライブラリからの情報の取得には対応できない．本機構は，パーミッションチェック機構にユーザが実行可否を選択する機構を実装している．Android では，個人情報を取得する API 呼出し時にはパーミッションチェックが行われる．そのため，パーミッションチェックのフックのみで個人情報の取得を制御することができ，Android への変更量を抑えられる．また，ネイティブライブラリからの API 呼出しの際にもユーザによる実行可否の選択が可能である．

また，Saint[9] は，アプリケーション開発者が，アプリ

ケーション間の通信の可否を制御するためのルールを定義することを可能にしている．アプリケーション開発者は，アプリケーション間通信が可能な条件として，端末が接続しているネットワークや場所などを指定することができる．Saint は，個人情報を持つアプリケーションを他のアプリケーションや信用できないネットワーク通信から保護すべきであるという観点から情報漏洩の対策を行っている．一方，本機構では，パーミッションが必要な API 呼出し時に実行可否をユーザが選択可能にするという対策を行っている．

TaintDroid[10] は，アプリケーションがいつ個人情報にアクセスし，どのように利用しているかをユーザに通知することを目的としている．Taint 解析によって個人情報を追跡し，外部に漏洩する際にユーザに通知する．利点は，アプリケーションやファイルを越えて個人情報が流出する場合でも検知が可能なことである．しかし，ネイティブコードを含む Shared Object ファイルを Taint 解析するのは困難なため，TaintDroid は Shared Object ファイルを含むアプリケーションが動作できないようにしている．

葛野 [11] は，パーミッションの組み合わせに着目し，情報ごとに送信先を確認し，情報の外部への送信の可否を制御する手法を提案している．この手法では，ネットワーク通信を監視し，個人情報が送信されていないかをアプリケーションによって確認しているため，Android のバージョンアップへの対応が容易であるという利点がある．しかし，アプリケーションが送信する個人情報を暗号化していた場合は検知が困難である．

松戸ら [12] は，インストール時にユーザにアプリケーションの危険性の注意を促すシステムを提案している．このシステムでは，ユーザがパーミッションを十分に理解できていないという観点からアプリケーションのインストール時にユーザに対して注意喚起を行う．一方，本機構ではアプリケーション開発者にユーザから適切な同意を得ることを強制するべきであるという観点から，アプリケーション動作時にユーザに同意を得ることを目的としている．

7. おわりに

本論文では，アプリケーションが個人情報を取得する時にユーザに通知し，同意を得る機構について述べた．本機構は，パーミッションチェック機構を拡張し，個人情報を取得する API 呼出しについてユーザが実行の可否を選択することを可能にする．また，パーミッション単位で既定値を設定することができる通知設定 AP を作成した．評価では，パーミッション動的制御機構によってアプリケーションによる API 呼出しの実行可否を選択できることを，広告が表示されるゲームアプリケーションを用いて確認した．

現在の実装では，同じアプリケーションからの個人情報の取得時に API 呼出しが複数回行われる場合に，ダイアログも複数回表示される．これにはユーザが選択した結果を一定時間キャッシュするという対策が考えられる．

参考文献

[1] Google Inc.: Android, `http://www.android.com` (2013).
[2] 総務省：スマートフォン プライバシー イニシアティブ–利用者情報の適正な取扱いとリテラシー向上による新時代イノベーション–，利用者視点を踏まえた ICT サービスに係る諸問題に関する研究会 (2012).
[3] Google Inc.: Android Emulator, `http://developer.android.com/tools/help/emulator.html` (2013).
[4] Google Inc.: Google Play, `http://play.google.com` (2012).
[5] Google Inc.: Noogra Nuts, `https://play.google.com/store/apps/details?id=com.bengigi.noogranuts` (2013).
[6] 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)，情報処理学会研究報告，Vol. 2013-CSEC-60, No. 57 (2013).
[7] 加藤真，松浦佐江子：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，情報処理学会研究報告，Vol. 2013-CSEC-61, No. 6 (2013).
[8] Google Inc.: Android NDK, `http://developer.android.com/tools/sdk/ndk/index.html` (2013).
[9] Ongtang, M., McLaughlin, S., Enck, W. and McDaniel, P.: Semantically Rich Application-Centric Security in Android, *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, Washington, DC, USA, IEEE Computer Society, pp. 340–349 (2009).
[10] Enck, W., Gilbert, P., Byung-Gon Chun, L. P. C., Jung, J., McDaniel, P. and Sheth, A. N.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pp. 1–6 (2010).
[11] 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，コンピュータセキュリティシンポジウム 2011 論文集，Vol. 2011, No. 3, pp. 155–160 (2011).
[12] 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, Vol. 2012-CSEC-56, No. 12, pp. 1–7 (2012).