SEAndroidの拡張によるAPの動的制御手法の提案

矢儀 真也　　山内 利宏

岡山大学大学院自然科学研究科
700-8530 岡山市北区津島中 3-1-1
yagi@swlab.cs.okayama-u.ac.jp, yamauchi@cs.okayama-u.ac.jp

あらまし Androidにおいて，管理者権限を取得して端末を制御するマルウェアやAPのパーミッションを悪用して個人情報を漏えいするマルウェアによる被害がある．これらのマルウェアへの対策として，SEAndroidを利用する方法がある．しかし，SEAndroidの機能を活用するには，ユーザがSEManagerを利用してAPのパーミッションを無効化する必要があるため，利用することは難しい．また，SEAndroidはポリシに従って，AP間の通信を制御するが，誤検知により必要な通信を拒否する可能性が高い．そこで，SEAndroidを拡張することにより，多くのユーザがAPを動的に制御できる手法を提案する．提案手法は，APがパーミッションを利用するとき，ユーザにパーミッションの利用の許可を自動的に求める．また，APが個人情報を漏えいする可能性のある通信をするとき，およびユーザがインストールしたAPが信頼できるAPと通信するとき，ユーザに通信の許可を求める．

Proposal of a Method for Dynamic Control of Application Programs by Extending the SEAndroid

Shinya Yagi　　Toshihiro Yamauchi

Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Kita-ku, Okayama, 700-8530, JAPAN

Abstract In Android, there is some damage by malicious software that obtains root privilege to control device or leaks personal information by using permission allowed to application programs (AP). SEAndroid is effective for mitigating damages caused by these malicious software. However, it is difficult for users to use SEAndroid properly. To use it, a user has to revoke permissions by SEManager. Moreover, policy enforcement in SEAndroid might reject required communication. In this paper, we propose a method for dynamic control of AP by extending SEAndroid. In our proposal, when AP uses a permission, our system automatically asks a user to allow it or not. Moreover, it also asks when there is a possibility that AP leaks personal information or AP installed by user communicates with trusted AP.

1　はじめに

近年，スマートフォンの普及が進んでいる．スマートフォン向けのオペレーティングシステムの1つとしてAndroid [1] がある．Androidの利用者は，Google Play [2] からアプリケーションプログラム（以降，APと略す）をスマートフォンに自由にダウンロードして，インストールできる．しかし，配布されているAPの中にはシステムの脆弱性を攻撃して，管理者権限を不正に取得するマルウェア，個人情報を外部に漏えいするマルウェア，およびネットワークやSMS経由で外部サーバと通信し，料金を発生させるマルウェアなどが発見され [3]，被害を及ぼしている．

上記のマルウェアによる被害は，管理者権限の問題とパーミッション機構の問題に分類できる．管理者権限を攻撃者が取得した場合，あらゆる操作が可能となる．また，APを利用するには，APのイン

ストール時に AP 作成者が要求するパーミッションをユーザが承認する必要がある．しかし，要求されたすべてのパーミッションをユーザが承認する必要があるため，ユーザにとって不要と考えられるパーミッションを不許可とすることができない．

Android のセキュリティ上の問題を改善するために，National Security Agency（以降，NSA と略す）[4] は Security Enhanced Android（以降，SEAndroid と略す）[5] を開発している．SEAndroid は Security Enhanced Linux（以降，SELinux と略す）[6] をベースとした強制アクセス制御（Mandatory Access Control : MAC）を持つ．これにより，特権を持つデーモンの動作を制限し，攻撃者による管理者権限の取得を防止できる [7]．また，AP に付与されているパーミッションを AP のインストール後に無効化できる Permission Revocation や，AP にタグを付与して，タグに基づいて AP 間の通信を制御する Tag Propagation と呼ばれる機能を持つ．

しかし，Permission Revocation を利用するには，ユーザが SEManager と呼ばれるツールを利用して AP ごとに不要なパーミッションを無効化する必要があるため，多くのユーザは利用することは難しい．また，Tag Propagation は誤検知により，必要な通信を拒否する可能性が高い．

これらの問題に対処するために，SEAndroid を拡張することにより，多くのユーザが SEAndroid の機能を利用して AP を動的に制御できる手法を提案する．提案手法は，AP がパーミッションを利用するとき，システムが自動的にユーザにパーミッションの許可を求める．これにより，ユーザがパーミッションを動的に制御できる．また，AP が個人情報を漏えいする可能性がある通信をするとき，およびユーザがインストールした AP が最初からインストールされている AP と通信をするとき，ユーザに通信の許可を求める．これにより，必要な通信を妨害することなく個人情報の漏えいを防止できる．

なお，ユーザビリティを考慮して，制御対象の AP をユーザが決定できることに加えて，一度ユーザが許可した操作は自動的に許可する．

2 Android のセキュリティと問題点

2.1 Android のセキュリティ

Android は Dalvik VM と呼ばれる仮想マシン上ですべての AP を実行する．AP は個別のプロセスとして動作するため，他の AP のメモリ空間にアクセスできない．また，AP ごとに固有のユーザ ID が割り当てられており，AP は他の AP の資源にアクセスできない．なお，AP が他の AP と通信するときはインテントを利用する．

AP がアクセスできる資源は，パーミッションで制御されている．たとえば，AP が電話帳にアクセスするには，READ_CONTACTS パーミッションが必要である．なお，AP が利用できるパーミッションは，AP のインストール時にユーザが承認する．

2.2 問題点

Android のセキュリティ上の問題点として，以下の 5 つがある [8]．

（問題 1）管理者権限の取得に関する問題

（問題 2）開発支援ツールに関する問題

（問題 3）パーミッションの悪用

（問題 4）情報漏洩の検知が不可能

（問題 5）WebKit の悪用

（問題 1）は管理者権限に関する問題であり，（問題 2），（問題 3），（問題 4），および（問題 5）はパーミッション機構に関する問題である．これらの問題は，SEAndroid を利用することで対処できる．

3 SEAndroid の機能と問題点

3.1 機能

SEAndroid は Android のセキュリティ上の問題を改善することを目的として，NSA が開発している．SEAndroid は SELinux をベースとして開発されており，SELinux の機能に加えて Android のパーミッションを制御する機能や AP 間の通信を制御する機能などがある．SEAndroid の機能を利用することで，管理者権限の取得や情報漏えいを防止できる．

SEAndroid はカーネルレベルのアクセス制御機構（以降，Kernel MAC と呼ぶ）と Android のフレームワークレベルのアクセス制御機構（以降，Middleware MAC と呼ぶ）をもつ．

Kernel MAC は特権を持つデーモンに固有のラベルを割り当てて，ラベルごとにアクセスできる資源を設定する．これにより，特権を持つデーモンの動作を制限し，攻撃者による管理者権限の取得を防止する．なお，管理者権限を取得できたとしても，Kernel MAC により，アクセスを制限できる．また，Multi Category Security を利用して AP ごとに違うカテゴリを設定し，AP 間の隔離を強制する．

Middleware MAC には以下の 3 つの機能がある．

(1) Install-time MAC

(2) Permission Revocation

(3) Tag Propagation

Install-time MAC は，AP のインストールを許可するか否かをポリシに基づいて判定する機能であり，マルウェアのインストールを防止できる．

Permission Revocation は，AP のインストール時にユーザが承認したパーミッションを，インストール後に無効化できる機能である．なお，パーミッションの無効化には SEManager を利用する．

Tag Propagation は，AP にタグを付与して，AP 間の通信によるタグの拡散を管理する機能である．たとえば，個人情報を取得するパーミッションを持つ AP に sensitive_data タグを付与する．この AP が他の AP と通信した場合，個人情報が拡散した可能性があるため，通信先の AP に sensitive_data タグを付与する．これにより，個人情報の拡散を管理できる．また，AP 間の通信をタグと関連付けたポリシに基づいて制御する．たとえば，個人情報を取得するパーミッションを持つ AP に sensitive_data タグを付与し，外部と通信するパーミッションを持つ AP に sinks タグを付与する．この 2 つの AP が通信する場合，個人情報が外部に漏えいする可能性があるため，通信を拒否する．これにより，個人情報の漏えいを防止できる．なお，タグは AP のインストール時，および AP 間の通信時に付与される．

3.2 問題点

（問題 1）最小特権のポリシの作成が困難

SEAndroid はプロセスなどの動作の主体（以降，サブジェクトと呼ぶ）とファイルなどの動作の対象（以降，オブジェクトと呼ぶ）にそれぞれラベルを割り当て，ラベルに基づいてホワイトリスト方式のアクセス制御を行う．つまり，ポリシで許可している動作は制限しないため，サブジェクトとオブジェクトのすべての組み合わせについて最小特権を満たすポリシを作成する必要がある．しかし，デーモンにはそれぞれ固有のラベル，AP には最初からインストールされている AP，ウェブブラウザ，およびユーザがインストールした AP の 3 種類ラベルがあることに加えて，オブジェクトには約 100 種類のラベルがあり，設定は難しい．以上のことから，最小特権のポリシを作成することは困難であるといえる．

（問題 2）SEAndroid の機能の活用が難しい

SEAndroid の機能である Permission Revocation を利用するために，ユーザは SEManager を利用して，AP ごとに不要なパーミッションを無効化する必要がある．しかし，ユーザは SEManager を知らないことや，どのパーミッションをあらかじめ無効化すればよいかわからないという問題がある．このため，多くのユーザは Permission Revocation を利用することは難しい．

（問題 3）必要な操作を拒否する可能性

SEAndroid の設定不備や Tag Propagation の誤検知により必要な操作を実行できない問題がある．Tag Propagation の誤検知の例として，カレンダ AP（com.android.calendar）がある．カレンダ AP は自身のプロバイダ（com.android.providers.calendar）と通信する．しかし，カレンダ AP は個人情報を取得するパーミッションを持ち，プロバイダはインターネットへアクセスするパーミッションを持つため，個人情報の漏えいの可能性を検知して通信を拒否する．これにより，必要な通信ができない問題がある．

本稿では，上記の問題のうち（問題 2），および（問題 3）の誤検知の問題に対処し，AP を動的に制御できる手法を提案する．

4 SEAndroid の拡張による AP の動的制御手法の設計

4.1 前提条件

提案手法では SEAndroid を利用する．このため，SEAndroid を利用できる Android 4.0.3 以降を対象とする．また，Kernel MAC のポリシは最小特権を実現しているものと仮定する．

4.2 設計方針

(1) ユーザが AP の動作を制御できること

(2) ユーザの判断を支援すること

(3) ユーザの負担を少なくすること

AP がパーミッションを利用するとき，および AP 間で通信するときに，AP の動作を許可するか否かをユーザに尋ねるように SEAndroid を拡張する．これにより，ユーザがパーミッションを動的に制御できるため，（問題 2）に対処できる．また，AP 間で通信するときに，通信を許可するか否かをユーザが決定できるため，Tag Propagation の誤検知による影響を緩和できる．これにより，（問題 3）の誤検知の問題に対処できる．

なお，AP の動作を制御するべきかどうかをユーザが判断できないことが考えられる．そこで，パー

ミッションの利用時と AP 間の通信時に，ユーザの判断を支援する情報をシステムが自動的に提示する．

また，すべての AP を制御対象とした場合，ユーザへの負担が大きいという問題がある．たとえば，ユーザが電話を掛けるたびに，電話を掛けることを許可するか否かをユーザが選択する必要がある．そこで，ユーザが制御対象の AP を決定できるようにすることで，ユーザの負担を減らす．さらに，一度ユーザが許可した操作に関しては自動的に許可することで，ユーザの負担を減らす．

4.3 設計

4.3.1 基本構成

本手法では AP によるパーミッションの利用を動的に制御する（以降，パーミッション制御機能と呼ぶ）．パーミッション制御機能は，AP が危険なパーミッションを利用するときに，利用の許可をユーザに求めることで，パーミッションを動的に制御する．これにより，(問題 2）に対処する．なお，ユーザにパーミッションの利用の許可を求めるとき，AP のコンポーネントの 1 つであるアクティビティが必要となる．このため，情報の提示やパーミッションの利用の許可をユーザに求める AP（以降，制御 AP と呼ぶ）を利用する．

また，AP ごとに付与しているタグを利用して，AP 間の通信を制御する（以降，通信制御機能と呼ぶ）．通信制御機能は，個人情報が漏えいする可能性がある通信をするとき，およびユーザがインストールした AP が最初からインストールされている AP と通信するときに通信の許可をユーザに求める．これにより，(問題 3）の誤検知の問題に対処する．

提案システムの全体像を図 1 に示す．図 1 の拡張した SEAndroid は，パーミッション制御機能と通信制御機能を示し，それぞれ SEAndroid の既存の機能である Permission Revocation と Tag Propagation を拡張したものである．なお，パーミッション制御機能は，パーミッションを制御するタイミング，制御対象のパーミッション，および制御対象の AP が Permission Revocation と異なる．また，通信制御機能は，タグの種類，制御対象の通信，およびタグを付与するタイミングが Tag Propagation と異なる．

これらの機能により，SEAndroid に詳しくないユーザでも危険なパーミッションを動的に制御できる．また，従来の問題である誤検知による通信拒否の影響を緩和する．さらに，個人情報の漏えいを防ぐことに加えて，最初からインストールされている AP の安全性を向上できる．

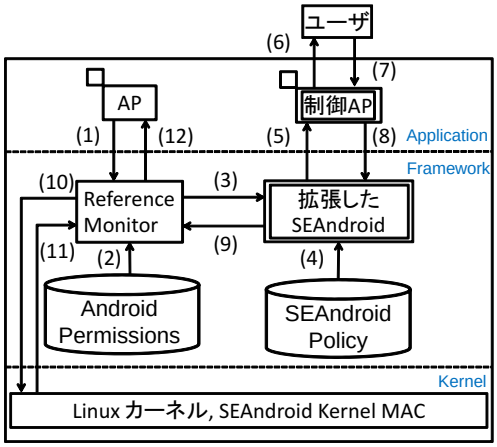

図 1 提案システムの全体像

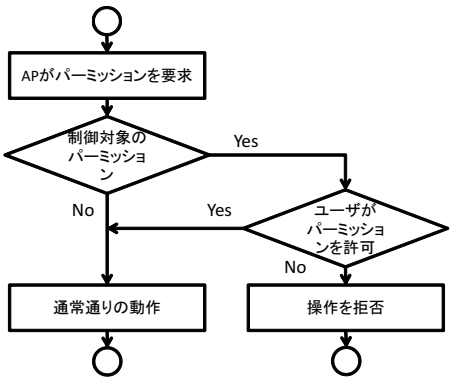

図 2 パーミッションの制御

4.3.2 パーミッション制御機能

パーミッション制御機能は AP によるパーミッションの利用を動的に制御する機能である．本機能は SEAndroid の既存の機能である Permission Revocation を拡張することにより実現する．本機能の処理の流れを図 2 に示し，以下で説明する．

(1) AP がパーミッションを要求

(2) システムが AP のタグ，および要求されたパーミッションを参照

(3) 制御対象の場合，ユーザにパーミッションの利用を許可するか否かを制御 AP を利用して尋ねる

(4) ユーザが許可した場合，パーミッションの利用を許可

(5) ユーザが許可しなかった場合，パーミッションの利用を不許可

本機能の設計として，制御対象のパーミッション，制御対象の AP，およびユーザに提示する情報を以下に記述する．

制御対象のパーミッション

すべてのパーミッションを制御対象とした場合，AP がパーミッションを要求するたびにユーザに許可を求めるため，ユーザの負担が大きい．そこで，制御対象のパーミッションを検討した．制御対象のパーミッションを表 1 に示す．なお，Android 4.0.4_r1.1（API Level 15）を対象として検討した．

制御対象のパーミッションを選択した基準は，パーミッションの保護レベルが dangerous であること，および個人情報を取得するパーミッション以外であることの 2 つである．

保護レベルは，normal，dangerous，signature，および signatureOrSystem の 4 つがある．normal は危険性が低いパーミッションである．dangerous は危険性が高いパーミッションであり，AP のインストール時にユーザが承認する必要がある．signature はパーミッションを定義している AP と利用する AP が同じ署名を持つ場合のみ利用できるパーミッションである．signatureOrSystem はシステムイメージに含まれている AP と同じ署名を持っている場合に利用でき，サードパーティの AP では利用されないパーミッションである．上記より，危険性が高い dangerous を対象とした．

また，個人情報を取得するパーミッションを対象外とした理由は，AP が個人情報を取得するだけでは，ユーザにとって不利益なことは起こりえないからである．このため，ユーザへの負担を考慮して対象外とした．

制御対象の AP

すべての AP を制御対象とした場合，ユーザの負担が大きい．そこで，制御対象の AP を検討した．

AP を最初からインストールされている AP（以降，trusted AP と呼ぶ），ユーザがインストールした AP のうちユーザが信頼できる AP（以降，user_trusted AP と呼ぶ），およびユーザがインストールした AP のうちユーザが信頼できない AP（以降，untrusted AP と呼ぶ）の 3 種類に分類する．

個人情報を外部に送信する可能性がある場合は，すべての AP を制御対象とし，表 1 のパーミッションを利用する場合は，untrusted AP を制御対象とする．これにより，trusted AP を利用して個人情報を外部に漏えいするマルウェア [9][10] に対処でき，ユーザの負担も少なくできる．

なお，user_trusted AP と untrusted AP は，AP を初めて起動したときにユーザが決定する．

表 1 制御対象のパーミッション

SIGNAL_PERSISTENT_PROCESSES
SET_ALWAYS_FINISH
SET_DEBUG_APP
SET_PROCESS_LIMIT
WRITE_CONTACTS
WRITE_HISTORY_BOOKMARKS
WRITE_SOCIAL_STREAM
ADD_VOICEMAIL
WRITE_CALENDAR
WRITE_PROFILE
SEND_SMS
CALL_PHONE
WRITE_SMS
BLUETOOTH
INTERNET
USE_SIP
NFC
AUTHENTICATE_ACCOUNTS
MANAGE_ACCOUNTS
USE_CREDENTIALS
WRITE_EXTERNAL_STORAGE
PROCESS_OUTGOING_CALLS
MODIFY_AUDIO_SETTINGS
CHANGE_WIFI_MULTICAST_STATE
CLEAR_APP_CACHE
PERSISTENT_ACTIVITY
DISABLE_KEYGUARD
BLUETOOTH_ADMIN
CHANGE_CONFIGURATION
MOUNT_UNMOUNT_FILESYSTEMS
CHANGE_WIMAX_STATE
SET_TIME_ZONE
MOUNT_FORMAT_FILESYSTEMS
WAKE_LOCK
WRITE_SYNC_SETTINGS
SYSTEM_ALERT_WINDOW
WRITE_SETTINGS
CHANGE_NETWORK_STATE
SET_ANIMATION_SCALE
REORDER_TASKS
CHANGE_WIFI_STATE
SUBSCRIBED_FEEDS_WRITE

ユーザに提示する情報

表 1 のパーミッションを利用するとき，システムがパーミッションの利用の許可をユーザに求める．しかし，パーミッション名だけではユーザは許可を求められているパーミッションにどのような危険性があるかを判断できない問題がある．

そこで，AP のパッケージ名，AP が利用するパーミッション名，およびパーミッションの内容を表示することでユーザの判断を支援する．

4.3.3 通信制御機能

通信制御機能は，AP 間の通信を動的に制御する機能である．本機能は SEAndroid の既存の機能である Tag Propagation を拡張することにより実現する．本機能の処理の流れを図 3 に示し，以下で説明する．

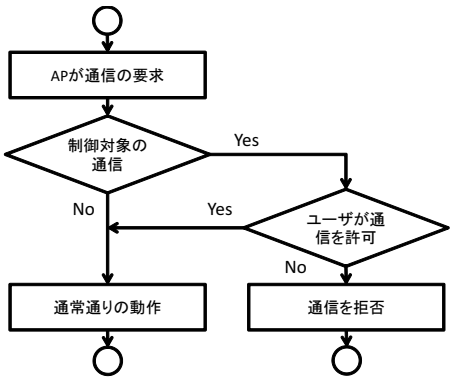

図 3 通信の制御

表 2 管理用のタグの一覧

trusted	最初からインストールされている AP
user_trusted	ユーザがインストールした AP のうちユーザが信頼できる AP
untrusted	ユーザがインストールした AP のうちユーザが信頼できない AP
launcher	AP を起動するランチャ（標準では com.android.launcher）

(1) AP が通信の要求

(2) システムが AP のタグを参照

(3) 個人情報が漏えいする可能性のある通信，または untrusted AP と trusted AP の通信の場合，ユーザに通信を許可するか否かを制御 AP を利用して尋ねる

(4) ユーザが許可した場合，通信を許可

(5) ユーザが拒否した場合，通信を拒否

本機能の設計として，タグの種類，制御対象の通信，タグを付与するタイミング，およびユーザに提示する情報を以下に記述する．

タグの種類

本機能では，AP に付与しているタグをもとに，AP 間の通信を制御するか否かを決定する．そこで，タグの種類を検討した．

タグは管理用のタグ（表 2）と情報漏えい防止用のタグ（表 3）の 2 種類に分類する．

管理用のタグは，trusted, user_trusted, untrusted, launcher の 4 つである．trusted は trusted AP に付与する．user_trusted は user_trusted AP に付与する．untrusted は untrusted AP に付与する．launcher はホーム画面である com.android.launcher に付与する．これらのタグを利用することで，AP の信頼度に基づいた制御ができる．なお，ランチャーは標準では com.android.launcher である．しかし，サードパーティのランチャーを利用することを考慮して launcher タグで管理する．

表 3 情報漏えい防止用のタグの一覧

sensitive_data	個人情報に関連するパーミッションに対応
sinks	情報を外部へ流出するパーミッションに対応

表 4 sensitive_data に関連付けるパーミッション

READ_SOCIAL_STREAM
READ_USER_DICTIONARY
READ_PROFILE
READ_HISTORY_BOOKMARKS
READ_LOGS
READ_CONTACTS
READ_CALENDAR
ACCESS_COARSE_LOCATION
ACCESS_FINE_LOCATION
READ_SMS
READ_ATTACHMENT
RECEIVE_SMS
RECEIVE_WAP_PUSH
RECEIVE_MMS
READ_PHONE_STATE
CAMERA
RECORD_AUDIO
GET_TASKS

情報漏えい防止用のタグは，sensitive_data と sinks の 2 つである．sensitive_data は個人情報の取得に関するパーミッションを対象とする．sensitive_data に関するパーミッションを表 4 に示す．パーミッションを選択した基準は保護レベルが dangerous であり，個人情報を取得できることである．sinks は個人情報を外部に漏えいする可能性があるパーミッションを対象とする．sinks に関するパーミッションを表 5 に示す．情報漏えい防止用のタグを利用して，個人情報の漏えいを防止する．

制御対象の通信

すべての AP の通信を制御した場合，ユーザの負担が大きい．たとえば，ユーザがホーム画面から AP を起動するときに AP の起動を許可するか否かを制御する．このため，制御対象の通信を検討した．

制御対象の通信は，sensitive_data を持つ AP と sinks を持つ AP の通信，および untrusted を持つ

表 5 sinks に関連付けるパーミッション

SEND_SMS
CALL_PHONE
INTERNET

AP と trusted を持つ AP の通信の 2 つとする．ただし，前者については，通信元と通信先の AP の両方が trusted を持つ場合，信頼できる動作と判断し，制御対象から除外する．これらの通信を制御することで，複数の AP で連携した個人情報の漏えい [9] の防止，および trusted AP の安全性を向上できる．

タグを付与するタイミング

SEAndroid は AP のインストール時に，AP に付与されているパーミッションに基づいてタグを付与する．このため，個人情報にアクセスしていない状態でも個人情報の漏えいの可能性を検知する問題がある．そこで，タグを付与するタイミングを検討した．

user_trusted と untrusted は AP の起動時にユーザが選択する．これにより，AP の起動時に AP を制御するか否かを決定できる．ユーザがインストールした AP は launcher タグを持つ AP から起動されるため，launcher タグを持つ AP からの通信をフックすることで実現できる．

sensitive_data はタグに関連付けられているパーミッションの利用時に付与する．これにより，個人情報の拡散の誤検知を削減できる．

上記以外のタグは，AP のインストール時に付与する．なお，タグは SEManager を利用して変更できるようにする．

ユーザに提示する情報

sensitive_data を持つ AP が sinks を持つ AP と通信するとき，または untrusted を持つ AP が trusted を持つ AP と通信するときに，システムがユーザに通信の許可を求める．しかし，通信を許可するか否かという情報だけではユーザが判断できない問題がある．そこで，通信元（先）AP のパッケージ名，通信元（先）AP のタグ名，および通信の内容を表示することでユーザの判断を支援する．

4.4 期待される効果

(1) ユーザが AP を安心して利用できること

(2) 個人情報の漏えいを防止

(3) 誤検知による通信の拒否の影響を緩和

(4) trusted AP の安全性を向上

パーミッションの利用時と AP 間の通信時にシステムが自動的にユーザに許可を求める．これにより，ユーザが AP の動作を制限できるため，ユーザが安心して AP を利用できる．また，個人情報が漏えいする可能性がある通信を検知するため，個人情報の漏えいを防止できる．さらに，Tag Propagation による誤検知が起こったとき，ユーザの判断で通信を許可するか否かを決定できる．これにより，誤検知による影響を緩和できる．加えて，untrusted AP が trusted AP と通信するときに，ユーザに通信の許可を求めるため，trusted AP の安全性を向上できる．

5 関連研究

個人情報の漏えいを防止する研究として，文献 [8]，[11]，[12]，および TaintDroid[13] がある．文献 [8] は，個人情報にアクセスする API と個人情報を外部に漏えいする API をフックして，ユーザが API の利用の可否決定をすることで個人情報の漏えいを防止する．文献 [11] は，AP に個人情報に対する参照ポインタのみを渡し，個人情報に関してはセキュリティマネージャ側で処理することで安全な個人情報の取り扱いを可能にするシステムを提案している．文献 [12] は，パーミッションの組み合わせや API の使用順番などをもとに個人情報が漏えいするパターンを作成し，外部との通信を制御する手法を提案している．この手法は，Linux カーネルを変更してパケットを監視することで実現されている．TaintDroid [13] は個人情報を保持する変数を追跡する機構であり，個人情報に関連付けられている変数が外部に漏えいする場合，検知する．

AP 間の通信を制御する研究として，IPC Inspection [14]，および文献 [9] がある．IPC Inspection はインテント経由の AP 間通信を制御する方式であり，通信先の AP が通信元の AP より多くの権限を持つ場合，通信先の AP の権限を通信元の AP の権限まで削減する方式である．これにより，複数の AP で連携して AP の権限を昇格する攻撃である confused deputy attacks に対処する．文献 [9] は，AP 間の通信の特徴をもとに作成した独自のポリシに基づいて，通信を制御する．この手法は，Android のフレームワークレベルの制御とカーネルレベルの制御を併用することで，インテント経由の通信だけでなく，ファイルシステムやネットワーク経由の通信を制御できることが特徴である．

AP の動作を許可するか否かをユーザが制御するという観点では，提案手法は文献 [8] と類似している．しかし，文献 [8] は個人情報にアクセスする API を対象としていることに対して，提案手法は危険なパーミッションを対象としている点，および制御対象の AP をユーザが選択できる点が異なる．

インテント経由の AP 間通信を制御するという観点では，提案手法は IPC Inspection と類似している．しかし，IPC Inspection は，通信元の AP の権限を通信先の AP が引き継ぐため，通信先の AP が必要な動作を実行できない可能性がある．提案手法は，AP の動作の制御はユーザが行い，システムがユーザの判断を支援する．これにより，必要な操作を実行できる点が異なる．

6 おわりに

SEAndroid の問題点として，最小特権のポリシの作成が困難であること，SEAndroid の機能の活用が難しいこと，および必要な操作を拒否する可能性があることを述べた．これらの問題に対処する方法として，SEAndroid の拡張による AP の動的制御手法を提案した．提案手法では untrusted AP がパーミッションを利用するとき，sensitive_data タグを持つ AP が sinks タグを持つ AP と通信するとき，および untrusted タグを持つ AP が trusted タグを持つ AP と通信するときにユーザに許可を求めることで，ユーザが AP を安心して利用できること，個人情報の漏えいを防止できること，誤検知による通信の拒否の影響を緩和できること，および trusted AP の安全性を向上できることを示した．

今後の課題として，提案システムの評価があげられる．評価項目は，従来の SEAndroid と提案手法をユーザへの負担の観点で比較すること，マルウェアによる個人情報の漏えいを防止する実例を示すこと，および誤検知の割合を測定することがあげられる．

謝辞 本研究の一部は，栢森情報科学振興財団平成 23 年度研究助成による．

参考文献

[1] “Android,” http://www.android.com/

[2] “Google Play,” https://play.google.com/store

[3] Y. Zhou, X. Jiang, “Dissecting Android Malware: Characterization and Evolution,” Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012), 2012.

[4] “National Security Agency,” http://www.nsa.gov/

[5] “SEAndroid - SELinux Wiki,” http://selinuxproject.org/page/SEAndroid

[6] “Security Enhanced Linux,” http://www.nsa.gov/research/selinux/

[7] “The Case for Security Enhanced (SE) Android,” Android Builders Summit 2012, 2012. https://events.linuxfoundation.org/images/stories/pdf/lf_abs12_smalley.pdf

[8] 奥田健嗣，中務　亮，山内利宏，“Android における情報伝搬の追跡と情報漏洩防止手法の提案，” 情報通信システムセキュリティ研究会（ICSS），電子情報通信学会研究報告，vol.111，no.495，pp.5-10，2012.

[9] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. Sadeghi, B. Shastry, “Towards Taming Privilege-Escalation Attacks on Android,” In Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS), 2012.

[10] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, “Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones,” In Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS), 2011.

[11] 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，“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，” 情報処理学会研究報告，vol.2012-CSEC-57，no.18，pp.1-6，2012.

[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，” コンピュータセキュリティシンポジウム（CSS2011）論文集，vol.2011，no.3，pp.155-160，2011.

[13] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones,” In 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2010.

[14] A. P. Felt, H. Wang, A. Moshchuk, S. Hanna, and E. Chin, “Permission re-delegation: Attacks and defenses,” In 20th USENIX Security Symposium, 2011.