6Z-6

Androidアプリケーション動作時に安全性を高める動的制御に関する検討

林　里香†　　後藤　厚宏†

情報セキュリティ大学院大学†

1. はじめに

近年，Android 搭載端末の普及が急速に進む一方で，端末内の情報の流出がしばしば取り沙汰され，安全性の確保が問題となっている．本稿では，端末利用時の安全性確保の一助となる，アプリケーション動作時の動的制御について検討する．

2. Androidのセキュリティ課題

Android では，アプリケーションの実行環境として，サンドボックス実行環境が提供されている．これにより，アプリケーションは，自分自身以外が作成したファイルの読み書き，他のプロセスへのアクセス，ネットワークアクセスや位置情報の取得など特定の機能の利用を制限されている．

アプリケーションがデフォルトで制限されている端末内の情報や機能を利用するためには，“パーミッション”が必要になる．パーミッションを付与されることで，アプリケーションはその範囲において，制限されていた情報や機能の利用が可能となる．

パーミッションは，アプリケーションの作成時に開発者の判断で使用が宣言され，インストール時にユーザが許可することで一括で付与される．この仕組みにより，パーミッションの許可／拒否はユーザに委ねられているが，インストール時に実際の動作（いつ，どのように，または，どのように組み合わされて利用されるのか）を把握することは困難であり，情報漏洩等の潜在的な危険性を認識することは難しい．加えて，インストール時以降，ユーザが，パーミッションやそれを利用したアプリケーションの挙動を把握・制御する手段は提供されていない．

以上をふまえ，情報漏洩のリスクを低減することを目的として，以下2点を主要な問題点とする．

(1) パーミッションがインストール時に一括付与され，項目ごとの許可／拒否ができない
(2) インストール時以降，ユーザが，パーミッションやそれを利用したアプリケーションの挙動を把握・制御することができない

これらの問題点に対し，以降の章では，アプリケーション動作時にその挙動を把握し，動的に制御する方法を検討する．

A Study on Android Security Improvement at Application Runtime
† Rika Hayashi and Atsuhiro Goto
† Institute of Information Security

3. Androidアーキテクチャの各層ごとの対策

Android は，Linux Kernel，Libraries，Android Runtime，Application Framework，Applications の5つのコンポーネントから成り，これらは階層構造を成している．

3-1. Application層での対策

Android では，各アプリケーションは個別のサンドボックス上で実行され，他のプロセスで動作するアプリケーションの挙動を制御することはできない．よって，Application 層のみで動的制御を実現することは難しい．

ただし，動的制御ではないが，制御対象となるアプリケーションのログの収集・解析により，以降のセキュリティの向上に役立てることは可能である．ログの解析から情報漏洩を検知する手法が提案されている[1]．

3-2. Linux Kernel層での対策

Linux Kernel 層に手を加えることにより，プロセス間通信の制御，外部送信パケットの制御等が可能となる．リアルタイムに端末内の重要情報を追跡し，情報漏洩を検知する仕組み[2]，外部ネットワークへのパケットの送信可否を制御する手法[3]が提案されている．

Linux Kernel 層における対策は，多様な手段をとることが可能となるが，既存システムとの親和性は低くなる．

3-3. Application Framework層での対策

Application Framework 層は，アプリケーションの実行に必要な各種 API を提供する．この API を改変することにより，任意の時点でのパーミッションの項目ごとの許可／拒否（問題点(1)の解消），及び，動的制御によるメソッド単位での許可／拒否や独自の制約の適用（問題点(2)の解消）が可能となる．

メソッドにフックすることで，独自の制御機構に制御を移し，これらを可能にする手法が提案されている[4][5]．

Application Framework 層における対策は，プロセス間通信の内容の取得や外部送信パケットのインターセプトができないことによる限界はある．しかし，既存システムとの親和性においては，Kernel 層での対策よりも優位に立つ．既存システムを生か

しつつ，動的制御を実現するには，Application Framework 層における対策が有効であると考える．

Application Framework 層における動的制御の流れを図 1 に示す．管理機能は，制御ポリシーの決定・施行機能，及び，これに必要な情報を保持する．制御対象アプリが端末内の情報へのアクセスを要求すると，メソッドに設けられたフックにより，制御が管理機能に移る(①)．管理機能では，制御ポリシーに基づき，要求の許可／拒否を判断する(②，③)．ここで，制御ポリシーが事前に設定されていない場合は，ユーザに問い合わせるか，インストール時の設定を適用する．要求許可ならば，端末内の情報へのアクセスが可能となり(④)，情報が取得される(⑤)．

この対策を実現する上での課題は，大きく以下の 2 点であると考える．

課題 A：端末内のリソースの制約への対処

課題 B：安全性と利便性のトレードオフ

B に関しては，特に，制御ポリシー決定における，安全性の確保とユーザへの負担軽減が重要である．

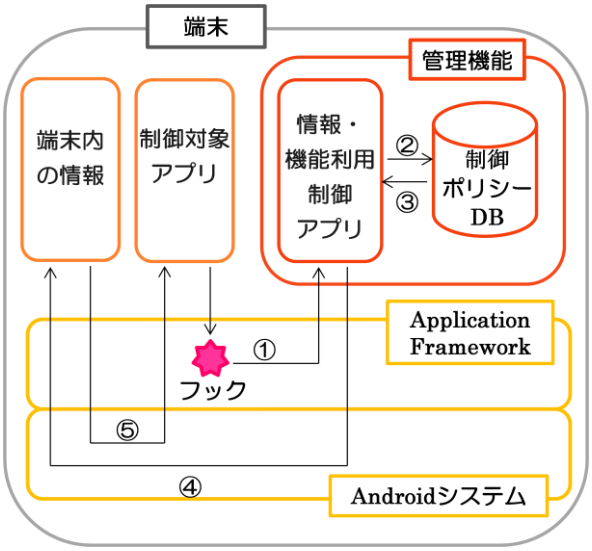

図 1 メソッドフック方式の概要

4. Cloud サービスによる強化

第 3 章で挙げた対策を強化するために，Cloud サービス（図 2）の活用を検討する． Cloud サービスを利用する目的は，以下の通りである．

- 豊富なリソースと情報を生かしてアプリケーションの挙動を詳細に解析する（課題 A への対処）
- 解析結果を端末内における制御ポリシーの決定に役立てることにより，安全性を確保しつつ，ユーザの負担を軽減する（課題 B への対処）

本 Cloud サービスが持つべき機能としては，アプリケーションレピュテーション情報の収集・分析機能，アプリケーション特性 DB の作成・管理機能がある．

端末から Cloud サービスへは，ログやアプリケーションに対する評価（ユーザ視点）情報を提供する．Cloud サービスから端末へは，ログの解析結果，制御ポリシー決定の材料となる情報を提供する．端末側管理機能は，Cloud サービスから提供された情報を解釈し，ユーザの判断と併せて制御に利用する．

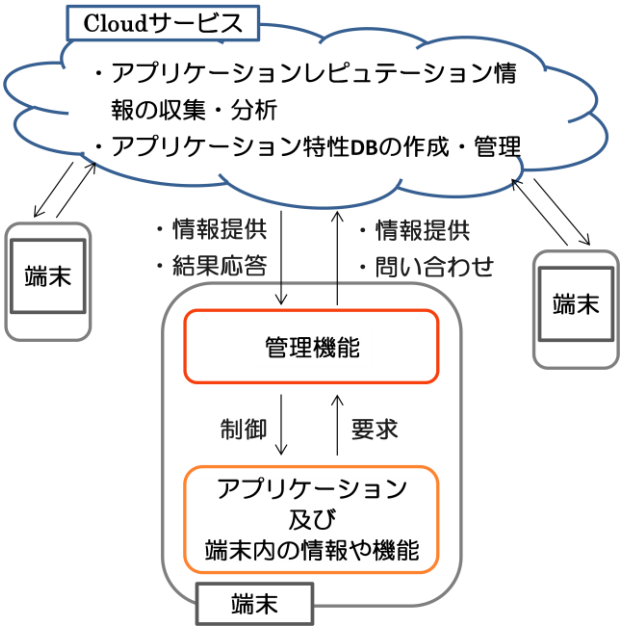

図 2 Cloud サービス概要

5. まとめ

今回は，既存の Android のセキュリティ機構の課題を示し，アプリケーション動作時にその挙動を把握し，動的に制御する方法を検討した．Application Framework 層における対策を実施することにより，前述問題点(1)は解消され，(2)に対しても有効な対策が可能であることを示した．また，その実現上の課題 A，B に対しても，Cloud サービスの運用方法，特に Cloud 側が提供する制御ポリシー決定材料の内容と端末側におけるその利用の仕方を工夫することで，一定の成果は得られると考える．

参考文献

[1] 竹森，他：Android 携帯電話上での情報漏洩検知，暗号と情報セキュリティシンポジウム，2011

[2] William, E., et al.: TaintDroid: An information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, 9th USENIX OSDI 2010

[3] 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，CSS 2011

[4] Mohammad, N., et al.: Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints, 5th ASIACCS 2010, ACM

[5] 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，CSS 2011