5E-6

スマートデバイスにおけるアプリケーション改竄検知方式に関する検討

飯塚智† 市原尚久† 山田達司†

株式会社 ＮＴＴデータ 技術開発本部 セキュリティ技術センタ†

1 はじめに

現在，スマートフォン等のスマートデバイスの OS やアプリケーション（以下，AP）の脆弱性に起因した情報漏洩等のセキュリティリスクが顕在化している．例えば攻撃者は，AP 開発者が開発した AP（以下，正規 AP）に対して「利用者がサーバから取得した個人情報を攻撃者サーバに送信する」等の攻撃用コードを埋め込むような改竄を行い，利用者の個人情報を不正に収集する．そのため， AP の完全性保証が重要となる．

AP の完全性保証のための従来技術としては，JDK™(Java Development Kit)付属の jarsigner 等を用いたコード署名を利用する方式[1]がある．この方式は，AP 開発時に jarsigner を用いて AP に含まれる個々のファイルのハッシュ値に対して署名を施し，AP のインストール時に行う署名検証により完全性を確認する．しかし，攻撃者は改竄後の AP に対して正規 AP の署名とは異なる署名を施すことにより，利用者は AP の改竄を検知できずに実行してしまうという問題がある．

本稿では，AP がサービスを提供するサーバ接続時に AP 改竄を検知可能な方式を提案する．

なお，本稿では，スマートデバイスの OS として Android™ OS を対象とする．

2 前提

本稿における AP 改竄の定義を以下に記述する．

[定義]攻撃者が，AP の一部を正規 AP と異なるコードに改変すること．

Android™ AP は Java 言語と C/C++言語の組合せによる開発が可能である．Java 言語の場合，コンパイル後の中間言語に対して容易に Reverse Engineering(RE)が行われ，AP 内部の処理ロジック等が解析される恐れがある．攻撃者は，RE を行うことにより，正規 AP の機能を実現し，任意の攻撃用コードを埋め込むことが可能となる．

C/C++言語の RE は，Java 言語のそれに比べて難易度が高いことが知られており，さらに C/C++言語で実装した Native コードに対して難読化処理を施すことにより RE の難易度を高めることが可能である．一方，C/C++言語 の場合，Java 言語に比べてメンテナンス性が低く，使用可能な Android™ の API が少ないという特徴がある．このため，C/C++言語により実装すべき機能以外は Java 言語を用いて実装すべきである．

本稿では，企業等が保有するサーバからサービスが提供され（以下，左記のサーバをサービスサーバと記載），利用者は AP を用いてサービスを利用し，サービスサーバに格納されている情報にアクセスするモデルを想定する．1 章で述べたようにサービスサーバが改竄された AP からのアクセスを許可した場合，情報漏洩等が発生する可能性がある．このため，改竄を検知し，検知結果に基づいてサービスサーバへの接続制限を行う等の対策を実施すべきである．

3 要件

改竄検知を行うためのサーバ（以下，検証サーバと記載）において AP の改竄を検知するためには，検証用データ(e.g. AP のハッシュ値)を保持する必要がある．更新頻度が高い AP のアップデート毎にアップデートに対応した検証用データを検証サーバに保持する場合，検証用データの更新頻度が高くなり，運用コストが高くなる．上記を考慮し，本稿の要件を以下に定義する．

(1)AP が改竄された場合には，Android™ 端末または検証サーバで検知可能であること．
(2)検証サーバに保持する AP 改竄検知のための検証用データの更新頻度が低いこと．
(3)AP とサービスサーバ間のデータを盗聴されても，再生攻撃をされないこと．

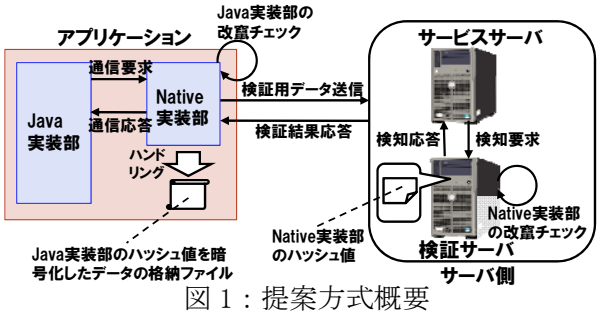

図 1：提案方式概要

A tamper detection method for application on smart devices
Satoshi IITSUKA†, Naohisa ICHIHARA†, Tatsushi YAMADA†
†NTT DATA CORPORATAION
Toyosu Center Bldg. Annex, 3-9, Toyosu 3-chome, Koto-ku, Tokyo 135-8671, Japan
iitsukas@nttdata.co.jp

(4)Android™ 端末をルート化し，各々の AP が保持するファイル等を操作されても，AP 改竄検知機能が，改竄された AP を検知可能なこと．
(5)Java 言語の RE により AP 改竄検知機能を容易に無効化されないこと．
(6)RE により C/C++言語で実装したセキュリティ上重要な機能を解析されないこと．

4　提案方式

4.1　提案方式概要

提案方式概要を図 1 に示す．本方式では，AP の機能を Java 言語と C/C++言語に分けて実装する．セキュリティ上重要な機能（e.g. AP 改竄検知機能，通信機能，暗復号機能）は C/C++言語により実装し，それ以外は Java 言語を用いて実装する．AP がサーバ側と通信する一連の処理において，Java 言語による実装部（Java 実装部），C/C++言語による実装部（Native 実装部）に対する改竄検知処理をそれぞれ Native 実装部，検証サーバで実行する．

本方式では， Java 実装部，Native 実装部に関するハッシュ値をそれぞれ Android™ 端末と検証サーバで保持し，それらに基づき改竄検知処理を実行することで要件(1)を満たす．Android™ 端末に保持するハッシュ値に対して暗号化を施すことで要件(4)を満たす．また，Native 実装部の機能は更新頻度が低くなるように設計し，Native 実装部のハッシュ値を検証用データとして検証サーバに保持することで要件(2)を満たす．さらに，サーバで生成した擬似乱数を利用し，Android™ 端末とサービスサーバ間のデータにワンタイム性を持たせることで要件(3)を満たす．本方式は，AP がサーバと通信するという一連の処理において，Native 実装部で AP 改竄検知機能を実行することにより要件(5)を満たす．Native 実装部に対して難読化を施すことで要件(6)を満たす．

4.2　提案方式詳細

提案方式は，(i)AP 開発フェーズ，(ii)サービス利用フェーズから構成される．

(i) AP 開発フェーズ

AP 開発者は AP 改竄検知機能やサーバとの通信機能等のセキュリティ上重要な機能を C/C++言語で実装し、それ以外のサービスを利用するための機能（以下，サービス利用機能と記載）等を Java 言語により実装することで AP を開発する．この際，暗号化用の共通鍵 key を生成し，Native 実装部に key をハードコーディングする．次に，AP の Java 実装部，および Native 実装部に対してそれぞれハッシュ計算を行う． Java 実装部のハッシュ値に対してのみ key を用いて暗号化を行い，AP に含まれる特定のファイルに保存する．さらに，Native 実装部に対して難読化を施す．AP 開発完了後に，AP 開発者は key と Native 実装部のハッシュ値を安全な方法で検証サーバに設定する．

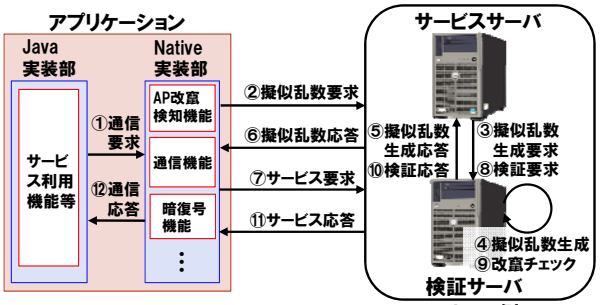

図 2：提案方式詳細

(ii) サービス利用フェーズ

本方式の詳細を図 2 に示す．AP 利用者は，インストールした AP を起動する．この際，Java 実装部は，Native 実装部に対して通信要求（e.g. Java 実装部での処理結果をサーバ側に送付するための要求）を行う（図 2：①）．通信要求を受けた Native 実装部は，まず，以下の手順を実行する．

1. 起動した AP の Java 実装部に対してハッシュ計算を行う．
2. AP 開発フェーズにおいて，AP の特定のファイルに暗号化して保存した Java 実装部のハッシュ値に対して key を用いて復号処理を行う．
3. 上記 1 と 2 のハッシュ値を比較し，一致する場合には次の処理へ進む．一致しない場合には AP を終了する．

次に，AP の Native 実装部に対してハッシュ計算を行う．以降，図 2 の②～⑪の処理は，Native 実装部とサーバ側の処理である．検証サーバで擬似乱数 Rand を生成し，Native 実装部では，上記で計算した Native 実装部のハッシュ値と Rand を結合したデータ列に対してハッシュ計算し，key を用いて暗号化する．検証サーバは，受信データに対して key を用いた復号処理と改竄チェックを実行する． Native 実装部は，サービスサーバからサービス応答（検証サーバで「改竄されていない」と判断した時の応答）を受信した場合，Java 実装部に通信応答を送付し（図 2：⑫），サービス利用機能を開始させる．それ以外の結果を受信した時は，AP を終了する．

5　考察とまとめ

本稿では AP がサーバの提供するサービスに接続する際に AP 改竄を検知可能な方式を提案した．提案方式は，3 章で述べた要件を満たす．

本方式の安全性は，Native 実装部の難読化に依存するため，今後は，難読化技術の検討や TPM(Trusted Platform Module)等の耐タンパハードウェアの導入が必要であると考える．

6　参考文献

[1] http://developer.android.com/guide/publishing/app-signing.html