「Android アプリのセキュア設計・セキュアコーディングガイド」サマリ

作　　成：　日本スマートフォンセキュリティ協会
　　技術部会　アプリケーションWG　セキュアコーディング グループ
ページ数：　232 頁(2012/06/01 版)
参加企業：　15 社（28 名）; Android セキュリティ部を含む; 2012/06/01 現在
目　　的：

現在、スマートフォンのアプリケーション開発者にはそれ相応の責任が生じている。従来の携帯電話ではあらかじめ課せられたセキュリティ制約によって、セキュリティについてあまり意識せずに開発したアプリケーションであっても比較的安全性が保たれていたが、スマートフォンでは携帯電話の重要な機能がアプリケーション開発者に解放されているため、アプリケーション開発者がセキュリティを意識して設計及びコーディングを行わなければ、スマートフォン利用者の個人情報が漏洩する、利用者の意図しない料金が発生するなどの危険性が高くなる。これは、例えば料金の発生する携帯電話機能をマルウェアに悪用される等の問題が生じる可能性が存在するからである。そこで、本グループでは Android アプリケーションのセキュア設計、セキュアコーディングのノウハウを集めて公開する事で、アプリケーション作成者のセキュアコーディングに対するスキルアップを第一の目的とし、それを通じてスマートフォンアプリケーションの安全性と一般ユーザーの安心感向上について寄与することを第二の目的とするものである。

文書範囲：

- この文書は一般の Android アプリ開発者に必要なセキュリティ Tips を集めることを目的とし、主にマーケット等で配布される Android アプリの開発におけるセキュリティ Tips が主なスコープとなる
- 初版では Java により実装する Tips だけを記載する。JNI 実装についても今後の版で記載していく予定である
- root 権限が奪取される脅威についても今のところ扱わない。基本的には root 権限が奪われていないセキュアな Android 端末を前提とし、Android OS のセキュリティモデルを活用したセキュリティ Tips をまとめる

影響範囲：

- ソフトウエアメーカ及び一般のアプリケーション開発者
- Android アプリケーション安全設計についての議論惹起
  - 話題になる事で一般にまで考え方が浸透し、底上げがなされると予想する

編集方針：

- タイムリーにノウハウを公開し共有していく事を第一とする
  - スマートフォンの置かれた環境が、フレームワークの増加・変化、ビジネス的な考え方によるコーディングの変化、新しいサービスの出現等、等により目まぐるしく変化するため
- 常にβ版のアプローチ
  - 最新かつその時点で正しいと思われることをできるだけ記載・公開し、間違いがあればフィードバックを頂いて常に正しい情報に更新し、タイムリーに提供する
  - 間違い・訂正箇所が発見された場合には、随時 JSSEC ホームページ等にて、訂正を行う（通

常の書籍などと同様）

- 成果を簡単に利用できる（開発者自身のものにできる）こと
  - 紹介しているコーディング方法について、文書内記載だけではなくサンプルコードを提供し、開発者が、プログラム中に取り込む、あるいは試行を行えるようにすることで、資料に掲載したセキュリティ情報を間単・確実に自分のノウハウとできること
- 編集ルール
  - ルールブックセクションでは、その記事がテーマとする開発者コンテキストにおいて、セキュリティ観点から守るべきルールや考慮事項を掲載する。ルールブックセクションの冒頭にはそのセクションで扱っているルールを表形式で一覧表示し、「必須」または「推奨」のレベル分けを行っている。
    - ✓ ルールには肯定文または否定文の 2 種類があり、必須の肯定文は「やらなきゃだめ」、推奨の肯定文は「やったほうがよい」、必須の否定文は「やったらだめ」、推奨の否定文は「やらないほうがよい」といったレベル感で表現している。
    - ✓ レベル分けは執筆者の主観に基づくものであり、参考として取り扱うことを読者に要求する。
  - アドバンストセクションでは、その記事がテーマとする開発者コンテキストにおいて、サンプルコードセクションやルールブックセクションで説明できなかった、しかし注意を要する事項について記載を行う。

文書利用許諾と提供物：

- 文書利用許諾（通常の書籍等と同様）
  - 文書には間違いが含まれている可能性があり、自己責任で利用すること
  - 文書に含まれる間違いを見つけた場合には連絡いただけること（お願いレベル）
    - 修正箇所については、随時ホームページ上に掲載する
  - 記述内容（プログラム含む）を公となるホームページ、書籍、文献等に記載する場合には出展を明記すること
- 提供物
  - 文書(PDF)：初回のみ印刷物としてイベントでの配布を検討中（1000 部程度）
  - サンプルコード
  - その他検討（電子ブック形式での配布等）
  - 電子的な提供物の配布に関しては、基本的に JSSEC のホームページにて行う

文書内容と構成：

１章　文書のねらい、及び利用許諾等について

２章　文書の構成について

３章　セキュアコーディングの基礎知識について

４章　安全にテクノロジーを活用する

(ア) Android で言えば Activity や SQLite など、テクノロジーごとにセキュリティ観点の癖というものがある。そうしたセキュリティの癖を知らずに設計、コーディングしていると思わぬ脆弱性をつくりこんでしまうことがある。この章では開発者が Android のテクノロジーを活用するシーンを想定した記事を扱う

(イ) Broadcast の送受信、SQLite の利用、ファイルの扱いなどについて

５章　　セキュリティ機能の使い方

(ア) 暗号や電子署名、Permission など、Android にはさまざまなセキュリティ機能が用意されている。これらのセキュリティ機能は取り扱いを間違えるとセキュリティ機能が十分に発揮されず抜け道ができてしまう。この章では開発者がセキュリティ機能を活用するシーンを想定した記事を扱う

(イ) パスワード入力画面を作る、Permission と Protection Level　などについて

内容についてのサンプル：

1.1. 入力データの安全性を確認する

入力データの安全性確認はもっとも基礎的で効果の高いセキュアコーディング作法である。プログラムに入力されるデータのうち、攻撃者が直接的、間接的にそのデータの値を操作可能であるものはすべて、入力データの安全性確認が必要である。以下、Activity をプログラムに見立て、Intent を入力データに見立てた場合を例にして、入力データの安全性確認の在り方について解説する。

Activity が受け取った Intent には攻撃者が細工したデータが含まれている可能性がある。攻撃者はプログラマが想定していない形式・値のデータを送り付けることで、アプリケーションに誤動作を誘発し、結果として何らかのセキュリティ被害を生じさせるのである。ユーザーも攻撃者の一人となり得ることも忘れてはならない。

Intent は action や data、extras などのデータで構成されるが、攻撃者が制御可能なデータはすべて気を付けなければならない。攻撃者が制御可能なデータを扱うコードでは、必ず次の事項を確認しなければならない。

(a) 受け取ったデータは、プログラマが想定した形式であって、値は想定の範囲内に収まっているか？
(b) 想定している形式、値のあらゆるデータを受け取っても、そのデータを扱うコードが想定外の動作をしないと保証できるか？

次の例は指定 URL の Internet 上の Web ページの HTML を取得し、画面上の TextView に表示するだけの簡単なサンプルである。　しかしこれには不具合がある。

Internet 上の Web ページの HTML を TextView に表示するサンプルコード

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    isr = new InputStreamReader(url.openConnection().getInputStream());
    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ...
```

(a)の観点で「urlstr が正しい URL である」ことを new URL()で MalformedURLException が発生しないことにより確認している。しかしこれは不十分であり、urlstr に「file://～」形式の URL が指定されると Internet 上の Web ページではなく、内部ファイルシステム上のファイルを開いて TextView に表示してしまう。プログラマが想定した動作を保証していないため、(b)の観点を満たしていない。

次は改善例である。(a)の観点で「urlstr は正規の URL であって、protocol は http または https に限定される」ことを確認している。これにより(b)の観点でも url.openConnection().getInputStream()では Internet 経由の InputStream を取得することが保証される。

Internet 上の Web ページの HTML を TextView に表示するサンプルコードの修正版

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    String prot = url.getProtocol();
    if (!"http".equals(prot) && !"https".equals(prot)) {
        throw new MalformedURLException("invalid protocol");
    }
    isr = new InputStreamReader(url.openConnection().getInputStream());
    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ...
```

入力データの安全性確認は Input Validation と呼ばれる基礎的なセキュアコーディング作法である。Input Validation という言葉の語感から(a)の観点のみ気を付けて(b)の観点を忘れてしまいがちである。データはプログラムに入ってきたときではなく、プログラムがそのデータを「使う」ときに被害が発生することに気を付けていただきたい。下記 URL もぜひ参考にしていただきたい。

- http://www.ipa.go.jp/security/awareness/vendor/programmingv2/clanguage.html
- http://www.jpcert.or.jp/java-rules/
- http://www.jpcert.or.jp/java-rules/android-j.html