Androidアプリの脆弱性を防ぐ、JSSECのセキュア設計・セキュアコーディングガイド

日本スマートフォンセキュリティ協会
技術部会　セキュアコーディンググループ

松並　勝 <Masaru.Matsunami@jp.sony.com>

安藤　彰 <Akira.Ando@jp.sony.com>

もくじ

1. Androidアプリ脆弱性の動向

2. Androidアプリ脆弱性事例とガイド

3. ガイド2012年11月1日版のトピック紹介

Androidアプリ脆弱性

動向

JAPAN SMARTPHONE SECURITY ASSOCIATION

Androidアプリの脆弱性報告が増加

- 2012年に入ってから急増化

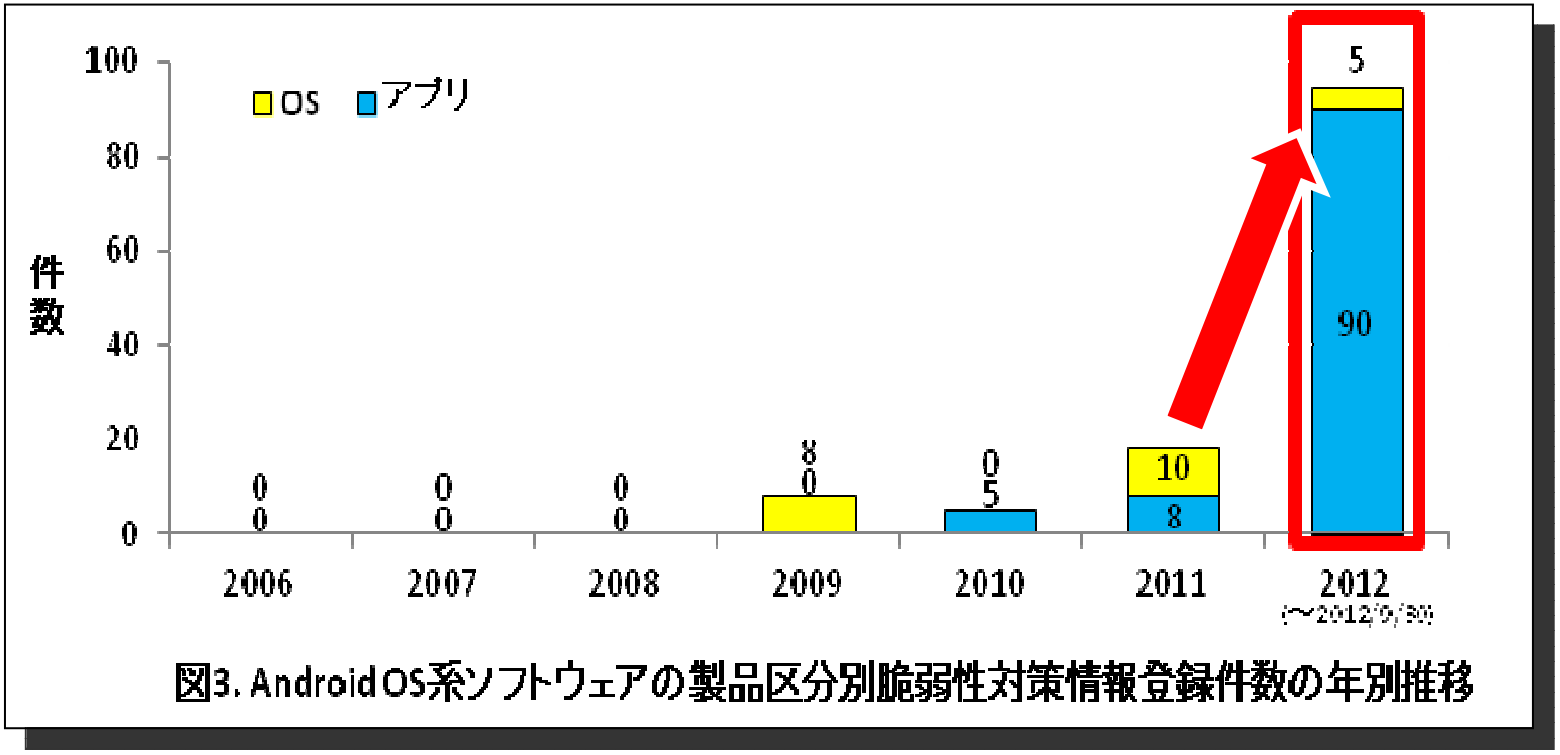


図3. Android OS系ソフトウェアの製品区分別脆弱性対策情報登録件数の年別推移

http://www.ipa.go.jp/security/vuln/report/JVNiPedia2012q3.html

セキュアコーディングガイド 公開

- 6月11日に公開
  - JSSECサイトに公開
    - ガイド文書（PDF）
    - サンプルコード一式（ZIP）
- 反響
  - 13万件超のダウンロード
  - 多数のWeb媒体で紹介
  - 複数の他言語翻訳オファー
  - 冊子も人気で増刷数回

ガイドの存在意義が高まってきた

3. IPA に報告された Android アプリの脆弱性

IPA は、「情報セキュリティ早期警戒パートナーシップ」において脆弱性関連情報の届出を受け付ける機関として活動している。本章では、IPA に届け出られた Android アプリの脆弱性の傾向を踏まえ、脆弱性が作り込まれた原因を考察する。

3.1. 脆弱性の傾向

2012 年 5 月末までに IPA に届け出られた Android アプリの脆弱性は累計 42 件である。これらの脆弱性の傾向を分析すると、「アクセス制限の不備」に起因するものが多いことが
3-1)。

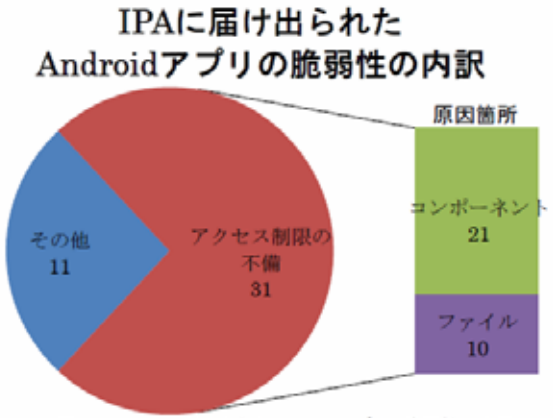


図 3-1 IPA に届け出られた Android アプリの脆弱性の内訳

http://www.ipa.go.jp/about/technicalwatch/pdf/120613report.pdf

- IPA報告の「アクセス制御の不備」はアプリコーディングが原因
- 対策方法としてJSSECガイド文書が紹介されている

本書は 5 つの章で構成している。Android アプリの脆弱性の傾向と対策を把握されたい方は、P.3 の「本書における用語」を確認した上で、3 章および 4 章を参照いただきたい。

章	章から分かること
1. Android アプリの概要	● Android アプリの特徴
2. Android の仕組み	● Android OS が提供する仕組みの概要
3. IPA に報告された Android アプリの脆弱性	● IPA に届け出られた Android アプリの脆弱性の傾向 ● 脆弱性が作り込まれた原因の考察
4. 脆弱性例の紹介	● Android アプリに作り込みやすい脆弱性と対策のポイント
5. 簡易チェックリスト	● 作り込みやすい脆弱性のチェックポイント

なお、本書では Android アプリの具体的なコーディングについては触れていない。コーディング例は、日本スマートフォンセキュリティ協会（JSSEC）が公開している『Android アプリのセキュア設計・セキュアコーディングガイド』[2]などをご参照いただきたい。

[1] 株式会社 MM総研： スマートフォン市場規模の推移・予測（12 年 3 月）
http://www.m2ri.jp/newsreleases/main.php?id=010120120313500 〔2012 年 5 月末日時点〕
[2] 『Android アプリのセキュア設計・セキュアコーディングガイド』【6 月 1 日版】
http://www.jssec.org/dl/android_securecoding.pdf 〔2012 年 6 月 12 日時点〕

2

Androidアプリ脆弱性

事例とガイド

JAPAN SMARTPHONE SECURITY ASSOCIATION

JVN (Japan Vulnerability Notes)

- JVNは脆弱性対策情報ポータルサイト
  - 脆弱性関連情報を提供
  - 対策情報を提供
- iPediaという脆弱性情報データベースがある
  - 「Android」で検索すると10月8日時点で138件の脆弱性が登録されていた

ほんとにあった Androidアプリの脆弱性

もちろん対策済み

いくつかのJVNの脆弱性を取り上げ、
次のような観点で解説して参ります。

- どのような脆弱性だったのか
- その原因は何だったのか
- どうあるべきだったのか
- ガイドには何と書いてあるのか

JAPAN SMARTPHONE SECURITY ASSOCIATION

JVN#31860555

事例1

JAPAN SMARTPHONE SECURITY ASSOCIATION

JVN#31860555
某twitter投稿アプリにおけるアクセス制限不備の脆弱性

ネットワークへの[illegible]限を持たないアプリ[illegible]の権限で画像を[illegible]

修正済み

Android用Twitterクライアント。
写真や動画の投稿が可能。

[illegible]ーザー
[illegible]してtweetす
[illegible]可能

公開日：20[illegible]

[illegible]限不備の脆弱性が存在します。

[illegible]の権限で画像ファイルをアップロードされる可能性があります。

ベンダ　　リンク
Tetsuya Aoyama　セキュリティアップデートのお願い
twicca - Google Play の Android アプリ

https://play.google.com/store/apps/details?id=jp.r246.twicca
http://jvn.jp/jp/JVN31860555/

JVN#31860555
某twitter投稿アプリにおけるアクセス制限不備の脆弱性
攻撃のシナリオ -個人情報流出-

https://play.google.com/store/apps/details?id=jp.r246.twicca

JVN#31860555
某twitter投稿アプリにおけるアクセス制限不備の脆弱性

攻撃のシナリオ -なりすまし投稿-

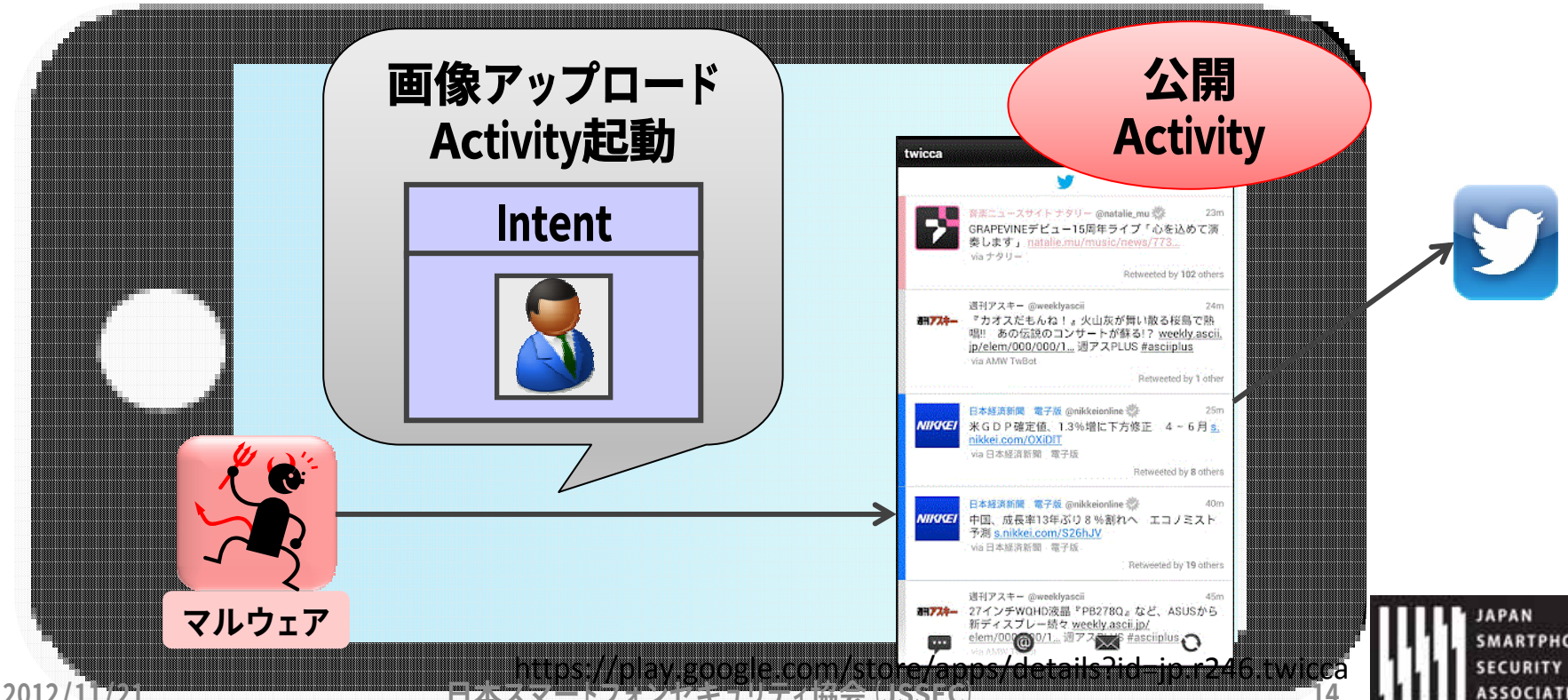
1. マルウェアが悪意のある画像のURL(file://...)を取得する
2. マルウェアがアプリの画像アップロード機能を呼び出してURLを渡す
3. アプリは指定された画像を添付してtweetする

https://play.google.com/store/apps/details?id=jp.r246.twicca

JVN#31860555
某twitter投稿アプリにおけるアクセス制限不備の脆弱性

脆弱性の原因

- アプリの画像アップロードActivityが他のアプリに公開されていたため、他のアプリから起動できる
- 画像アップロードActivityはアプリ内部のみでの使用が想定されていた



JVN#31860555
某twitter投稿アプリにおけるアクセス制限不備の脆弱性
対策

アプリの画像アップロードActivityを他のアプリに対して公開しないように設定する

AndroidManifest.xml

```
...
<activity
  android:name=".画像アップロードActivity"
  ...
  android:exported="false" />
...
```

https://play.google.com/store/apps/details?id=jp.r246.twicca

JVN#31860555
某twitter投稿アプリにおけるアクセス制限不備の脆弱性

セキュアコーディングガイドを参照する

ポイント
アプリを実装する際のポイント

サンプルコードポイントを実装したアプリのサンプルコード
•マニフェストファイル
•javaファイル

アプリ内部でのみ使用するActivityを扱う

exported="false"により、明示的に非公開設定する

android:exported="false"

JVN#67435981

事例2

JAPAN SMARTPHONE SECURITY ASSOCIATION

JVN#67435981
某リアルタイムコミニュケーションアプリにおける暗黙的 Intent の扱いに関する脆弱性

暗黙的Intentの扱い[illegible]切で、送信したメッセージ情報[illegible]から読み取られる

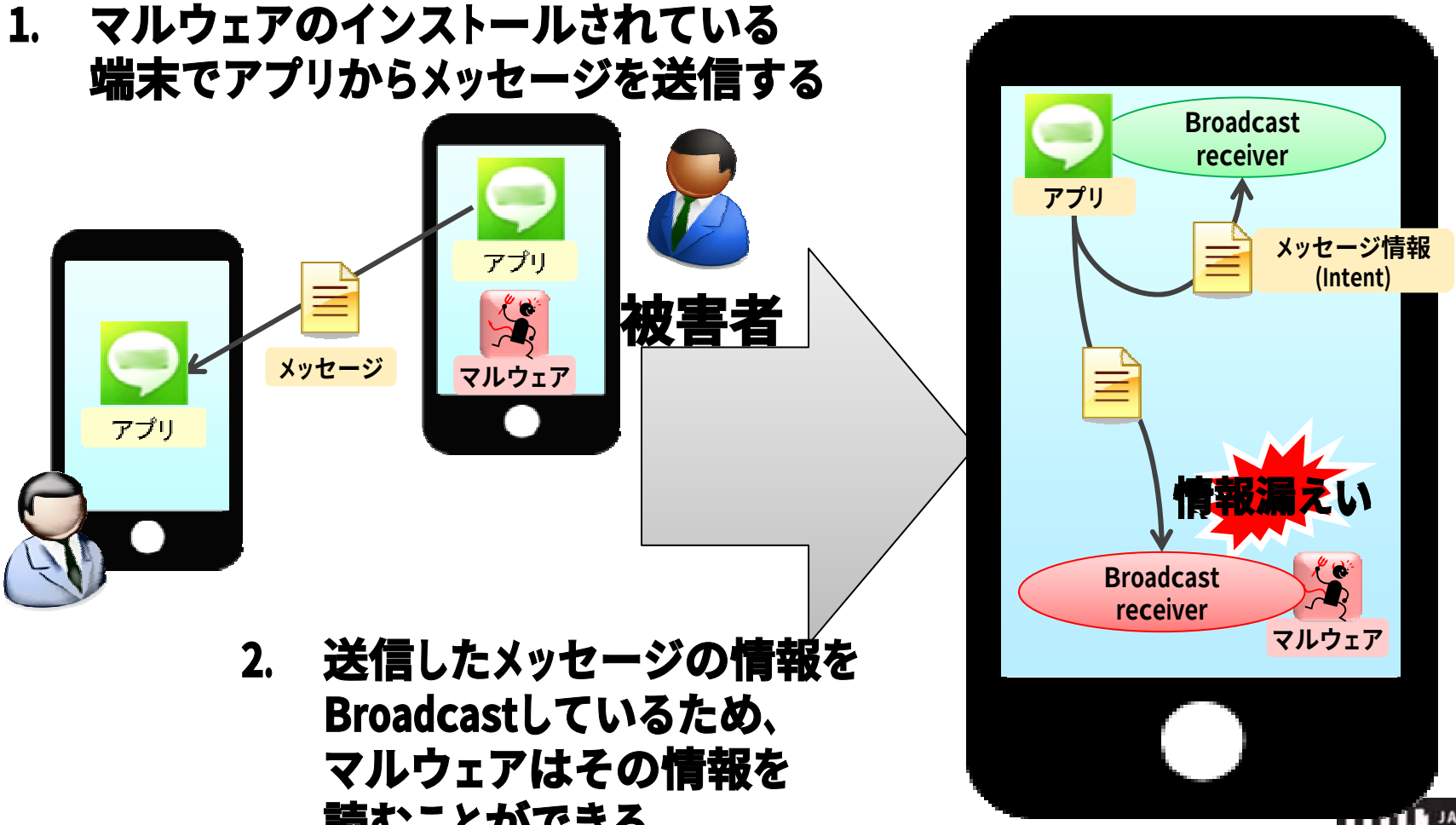
[illegible]ト電話やテキスト[illegible]などでコミュニケーショ[illegible]行う。絵文字の豊富さなど[illegible]で人気を集めている。

修正済み

https://play.google.com/store/apps/details?id=jp.naver.line.android&hl=ja
http://jvn.jp/jp/JVN67435981/

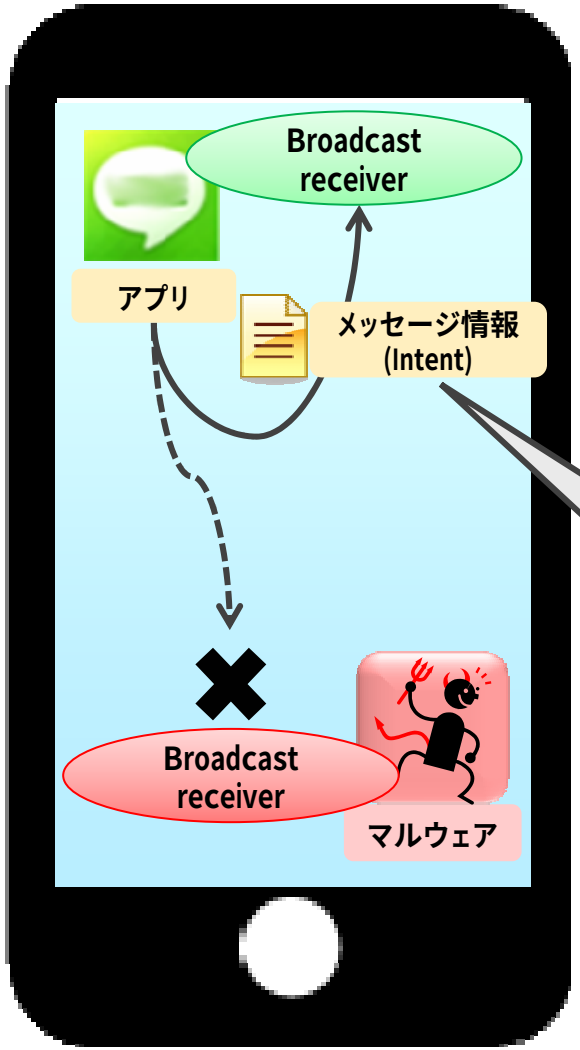
攻撃のシナリオ

1. マルウェアのインストールされている端末でアプリからメッセージを送信する



2. 送信したメッセージの情報をBroadcastしているため、マルウェアはその情報を読むことができる

対策



Q. どうするべきだったのか？

A. 明示的Intentを使う

自アプリ内のBroadcast receiverに送信する目的であれば、明示的Intentにより宛先クラスを限定するべき。

明示的Intentにより、Broadcast送信する宛先を自アプリのコンポーネントに固定

セキュアコーディングガイドを参照する

4.2.1.1. 非公開Broadcast Receiver - Broadcast を受信する･送信する

ポイント(Broadcast を送信する):

4. 同一アプリ内 Receiver はクラス指定の明示的 Intent で Broadcast 送信する
5. 送信先は同一アプリ内 Receiver であるため、センシティブな情報を送信してよい
6. 同一アプリ内 Receiver からの結果情報であっても、受信データの安全性を確認する

PrivateSenderActivity.java

```
package org.jssec.android.broadcast.privatereceiver;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class PrivateSenderActivity extends Activity {

    public void onSendNormalClick(View view) {
        // ★ポイント4★ 同一アプリ内 Receiver はクラス指定の明示的 Intent で Broadcast 送信する
        Intent intent = new Intent(this, PrivateReceiver.class);

        // ★ポイント5★ 送信先は同一アプリ内 Receiver であるため、センシティブな情報を送信してよい
        intent.putExtra("PARAM", "センシティブな情報 from Sender");
        sendBroadcast(intent);
    }

    public void onSendOrderedClick(View view) {
```


同一アプリ内Receiverにはクラス指定の明示的Intentで Broadcast送信する

JVN#92038939

事例3

JAPAN SMARTPHONE SECURITY ASSOCIATION

JVN#92038939
某SNSアプリにおける情報管理不備の脆弱性

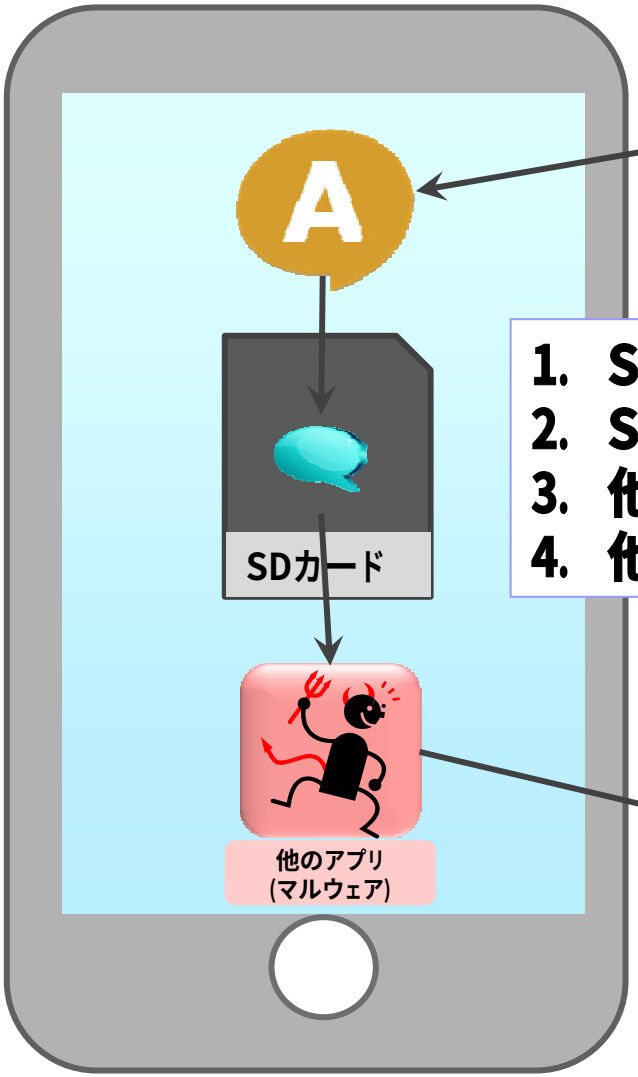
他のアプリが[illegible]言」の内容を取得[illegible]

修正済み

[illegible]きの閲覧/投稿/削除、
[illegible]人の更新情報の閲覧などの
機能が利用可能

https://play.google.com/store/apps/details?id=jp.mixi
http://jvn.jp/jp/JVN92038939/

攻撃のシナリオ



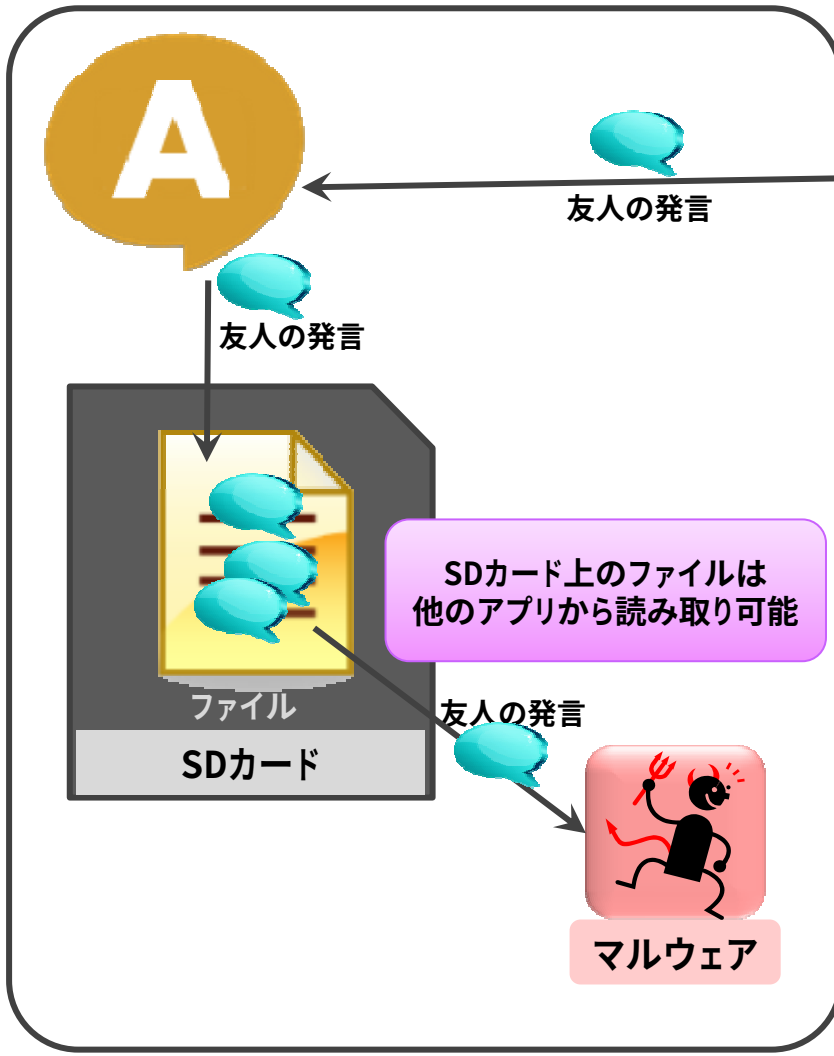
友人の発言

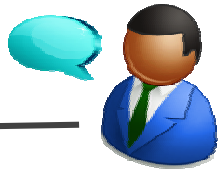
1. SNSアプリが友人の発言を取得
2. SNSアプリが友人の発言をSDカードに保存
3. 他のアプリが発言をSDカードから取得
4. 他のアプリが発言を攻撃者に渡す

個人情報
流出

攻撃者

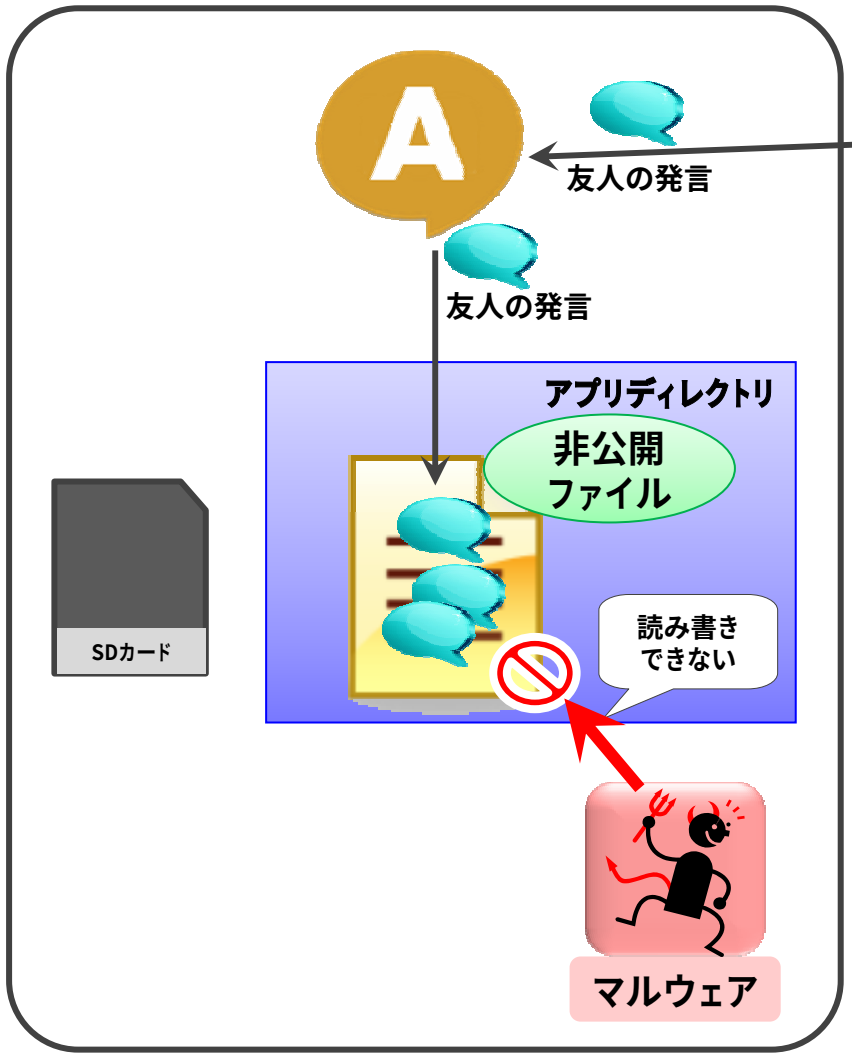
脆弱性の原因





- 友人の発言の保存先がSDカードになっていた
- SDカードに保存されたファイルには、他のアプリによる読み取りが可能

対策



「友人の発言」を非公開ファイルに保存する

JVN#92038939
某SNSアプリにおける情報管理不備の脆弱性

セキュアコーディングガイドを参照する

4.6.1.1. 非公開ファイルを扱う

ポイント:

1. ファイルは、アプリディレクトリ内に作成する
2. ファイルのアクセス権は、他のアプリが利用できないようにプライベートモードにする
3. センシティブな情報を格納することができる
4. ファイルに格納する(された)情報に対しては、その入手先に関わらず内容の安全性を確認する

```
/**
 * ファイルの作成処理
 *
 * @param view
 */
public void onCreateFileClick(View view) {
    FileOutputStream fos = null;
    try {
        // ★ポイント1★ ファイルは、アプリディレクトリ内に作成する
        // ★ポイント2★ ファイルのアクセス限は、他のアプリが利用できないようにプライベートモードにする
        fos = openFileOutput(FILE_NAME, MODE_PRIVATE);

        // ★ポイント3★ センシティブな情報を格納することができる
        // ★ポイント4★ ファイルに格納する情報に対しては、その入手先に関わらず内容の安全性を確認する
        // サンプルにつき割愛。「3.2 入力データの安全性を確認する」を参照。
        fos.write(new String("センシティブな情報(File Activity)¥n").getBytes());
    } catch (FileNotFoundException e) {
```

同一アプリ内でのみ読み書きするファイルは openFileOutput()でMODE_PRIVATEを 指定して作成する

JAPAN SMARTPHONE SECURITY ASSOCIATION

JVN#23328321

事例4

JAPAN SMARTPHONE SECURITY ASSOCIATION

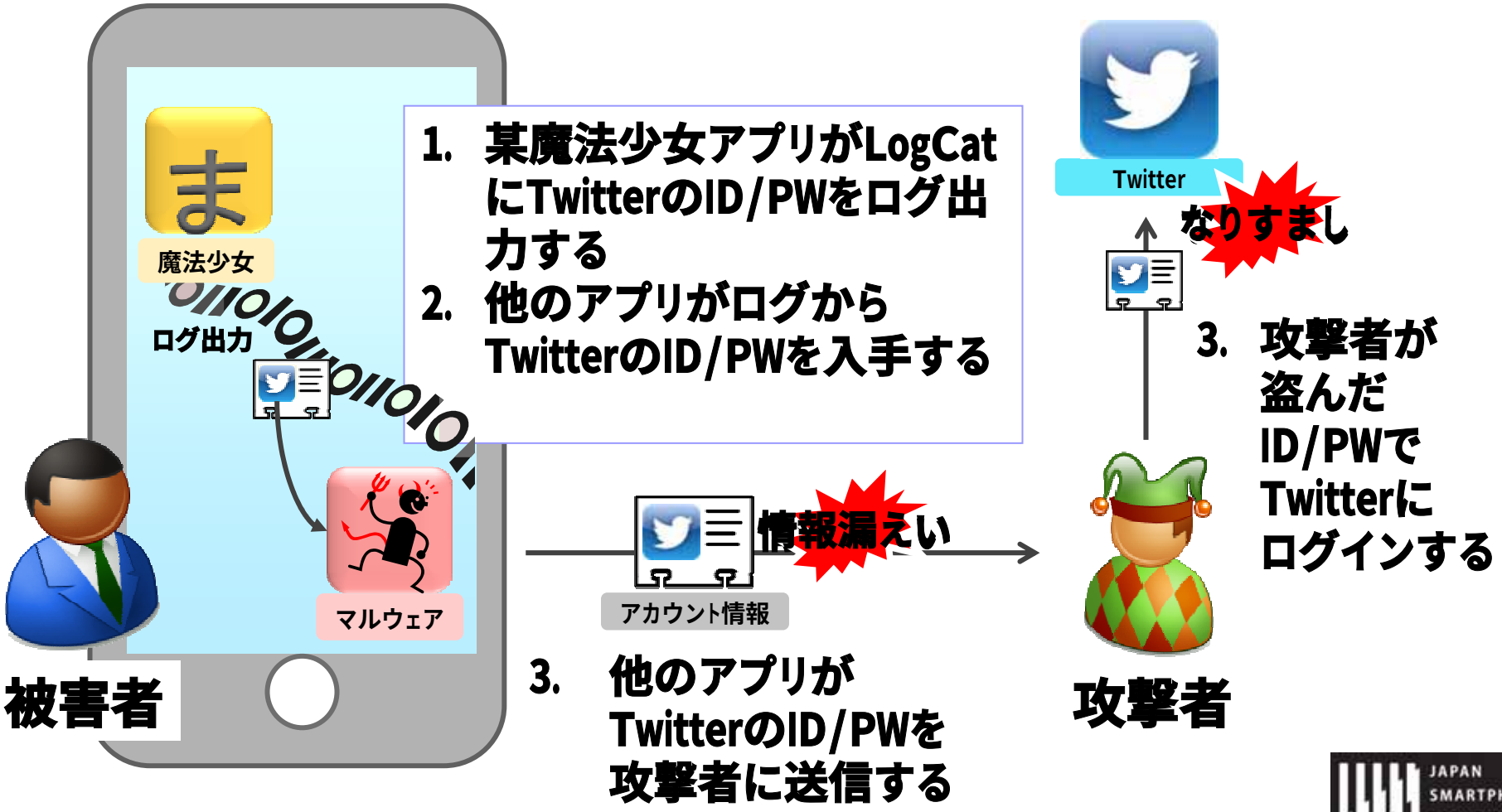
JVN#23328321
某魔法少女アプリ における情報漏えいの脆弱性

このアプ Twitter連携機 のアプリ PWを読み できる

修正済み

http://mad pplication
http://jvndb -2012-000054.html
http://jvn.jp/jp

攻撃のシナリオ


1. 某魔法少女アプリがLogCatにTwitterのID/PWをログ出力する
2. 他のアプリがログからTwitterのID/PWを入手する
魔法少女
ログ出力
マルウェア
被害者
Twitter
なりすまし
3. 攻撃者が盗んだID/PWでTwitterにログインする
情報漏えい
アカウント情報
3. 他のアプリがTwitterのID/PWを攻撃者に送信する
攻撃者

JVN#23328321
某魔法少女アプリにおける情報漏えいの脆弱性

脆弱性 の 原因 と 対策

原因:

- アプリ開発中のデバッグ用にTwitterのID/PWをログ出力していたと思われる
- ログ出力したままアプリをリリースしてしまったと思われる
- READ_LOGS permissionを利用宣言している他のアプリはLogCatに出力されたログを読み取ることが可能

対策:

リリースビルドではデバッグログを
LogCatへ出力しないようにする

JVN#23328321
某魔法少女アプリにおける情報漏えいの脆弱性

セキュアコーディングガイドを参照する

LogCatにログを出力する

ポイント:

1. ログの出力方法を統一する
2. センシティブな情報を含むログはデバッグビルドの時のみ出力する
3. 自前の Log クラスを作成する

UseLogActivity.java

```
package org.jssec.android.util.log;

import android.app.Activity;
import android.os.Bundle;

public class UseLogActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_output_log);

        // ★ポイント1★ ログの出力方法を統一する
        // 自前のLogクラスを使用してログを出力する
        Log.d("tag", "message");
    }
```

Log.java

```
package org.jssec.android.util.log;

// ★ポイント3★ 自前のLogクラスを作成する
public class Log {
    public static final int VERBOSE = android.util.Log.VERBOSE;
    public static final int DEBUG = android.util.Log.DEBUG;
    public static final int INFO = android.util.Log.INFO;
    public static final int WARN = android.util.Log.WARN;
    public static final int ERROR = android.util.Log.ERROR;
    public static final int ASSERT = android.util.Log.ASSERT;

    public s
        retu
    }

    public s
        retu
    }

    public static int d(String tag, String msg) {

      // ★ポイント2★ センシティブな情報を含むログはデバッグビルドの時のみ出力する
        if (SecApplication.isInDebugMode()) {
            return android.util.Log.d(tag, msg);
        } else {
            return -1;
        }
    }
```

次版掲載予定

センシティブな情報を含むログはデバッグビルドの時のみ出力する

JAPAN SMARTPHONE SECURITY ASSOCIATION

OSVDB#74648

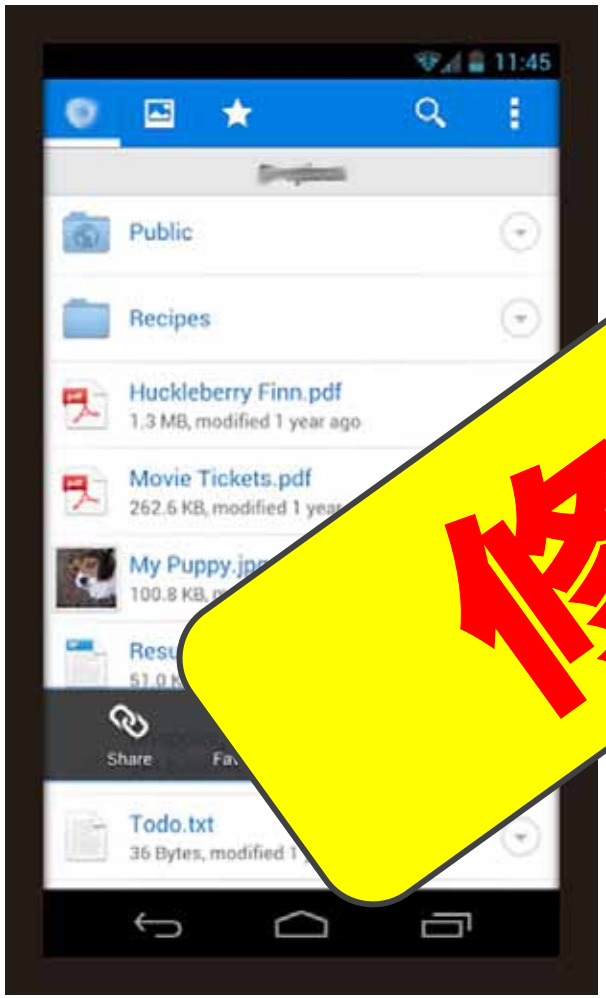
事例5

JAPAN SMARTPHONE SECURITY ASSOCIATION

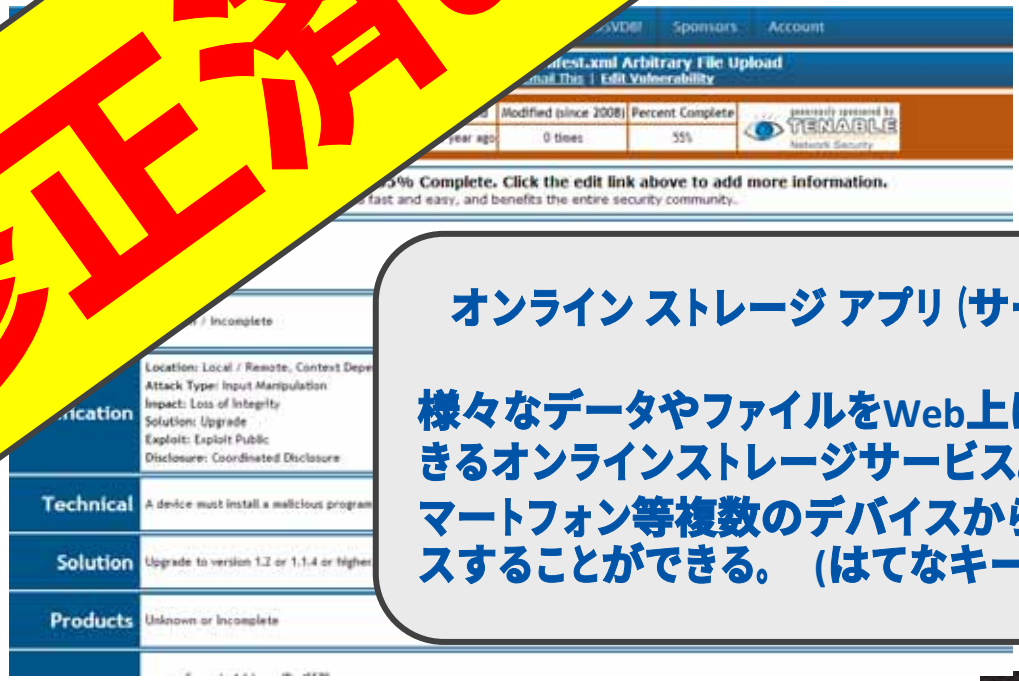
OSVDB#74648

某オンラインストレージアプリにおける ContentProviderアクセス範囲 弱性

他のアプリが セスできる
任意の のストレージに



修正済み

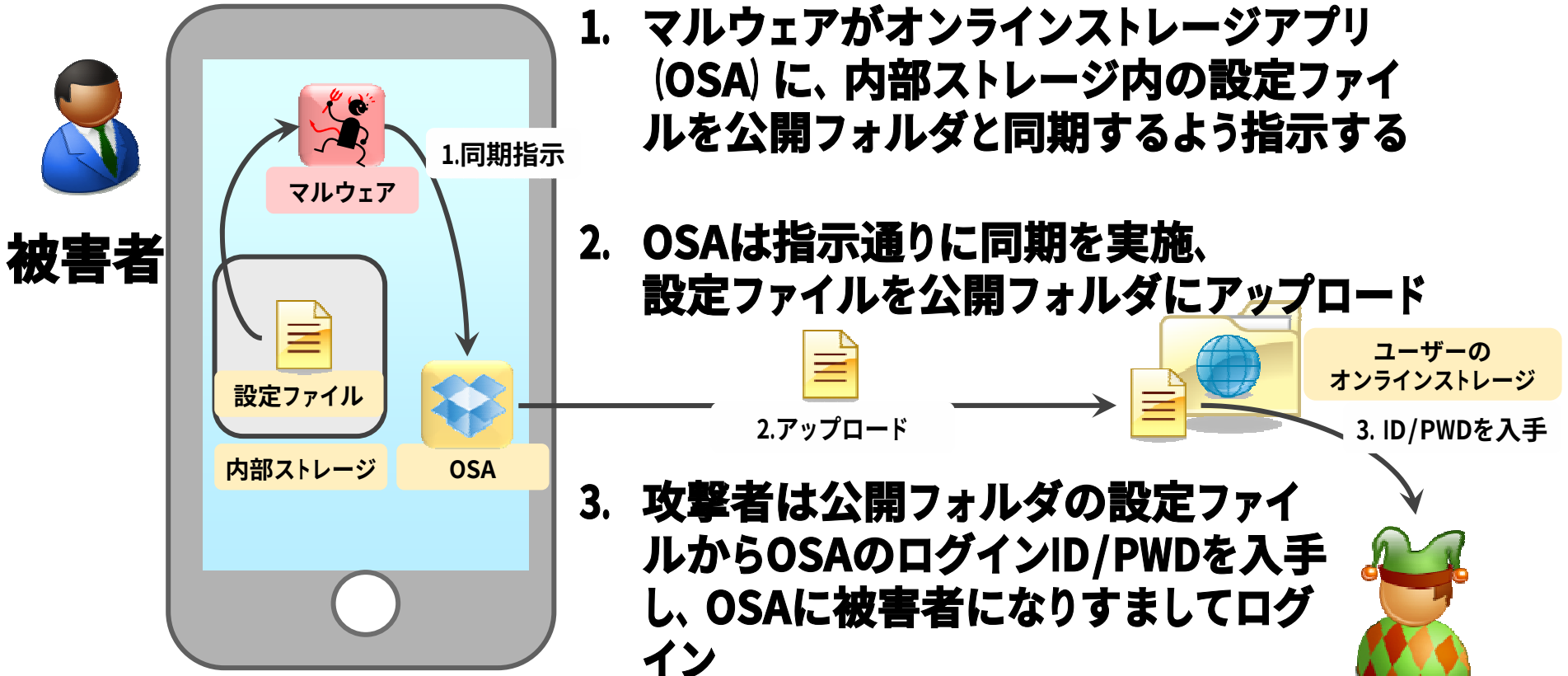


オンライン ストレージ アプリ (サービス)

様々なデータやファイルをWeb上に保存できるオンラインストレージサービス。PCやスマートフォン等複数のデバイスからアクセスすることができる。 (はてなキーワード)

https://play.google.com/store/apps/details?id=jp.r246.twicca
http://osvdb.org/show/osvdb/74648

攻撃のシナリオ –アカウント乗っ取り-

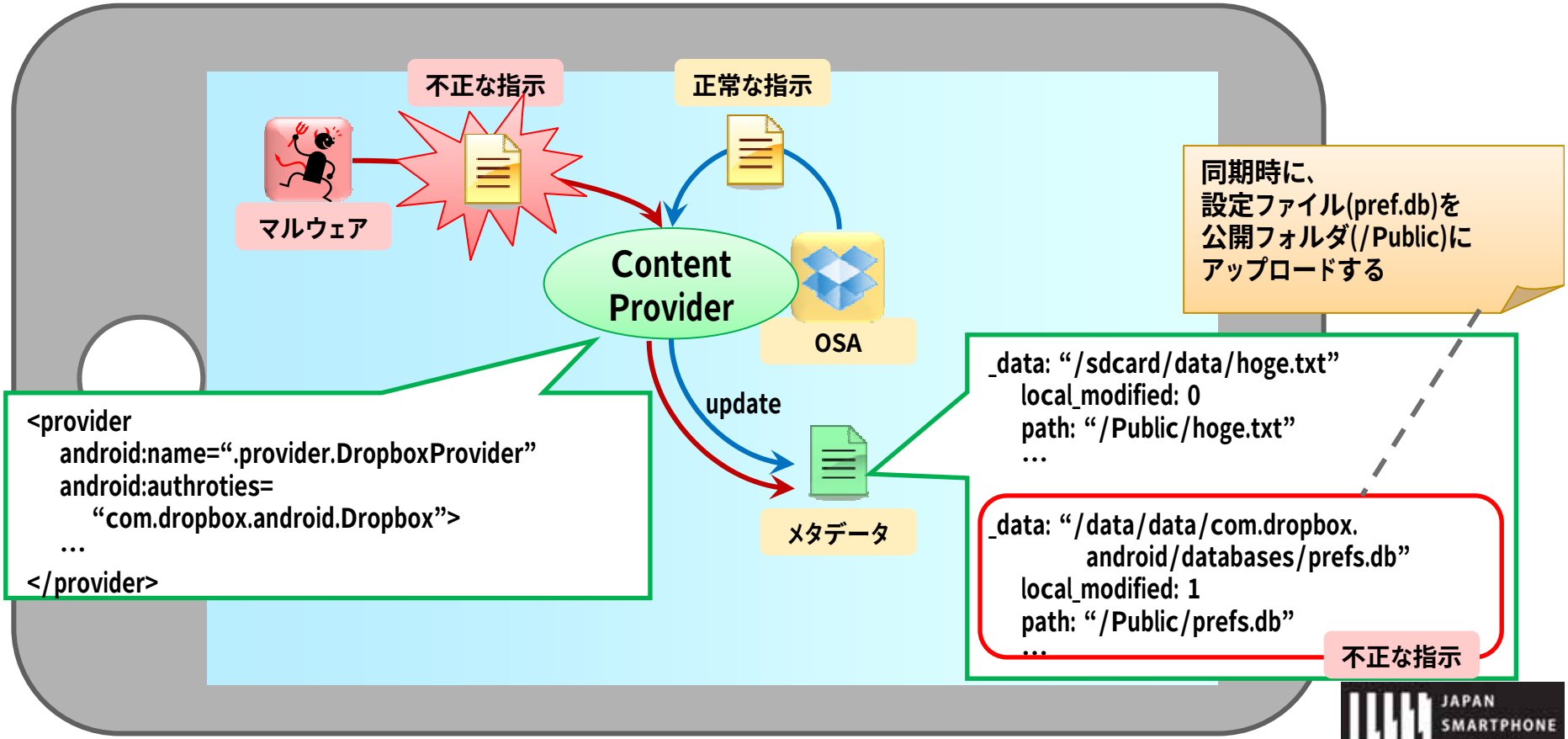


1. マルウェアがオンラインストレージアプリ（OSA）に、内部ストレージ内の設定ファイルを公開フォルダと同期するよう指示する
2. OSAは指示通りに同期を実施、設定ファイルを公開フォルダにアップロード
3. 攻撃者は公開フォルダの設定ファイルからOSAのログインID/PWDを入手し、OSAに被害者になりすましてログイン
4. 攻撃者は非公開フォルダのファイルにアクセスしたり、アカウントの設定を変えたり…

OSVDB#74648
某オンラインストレージアプリにおける
ContentProviderアクセス範囲の脆弱性

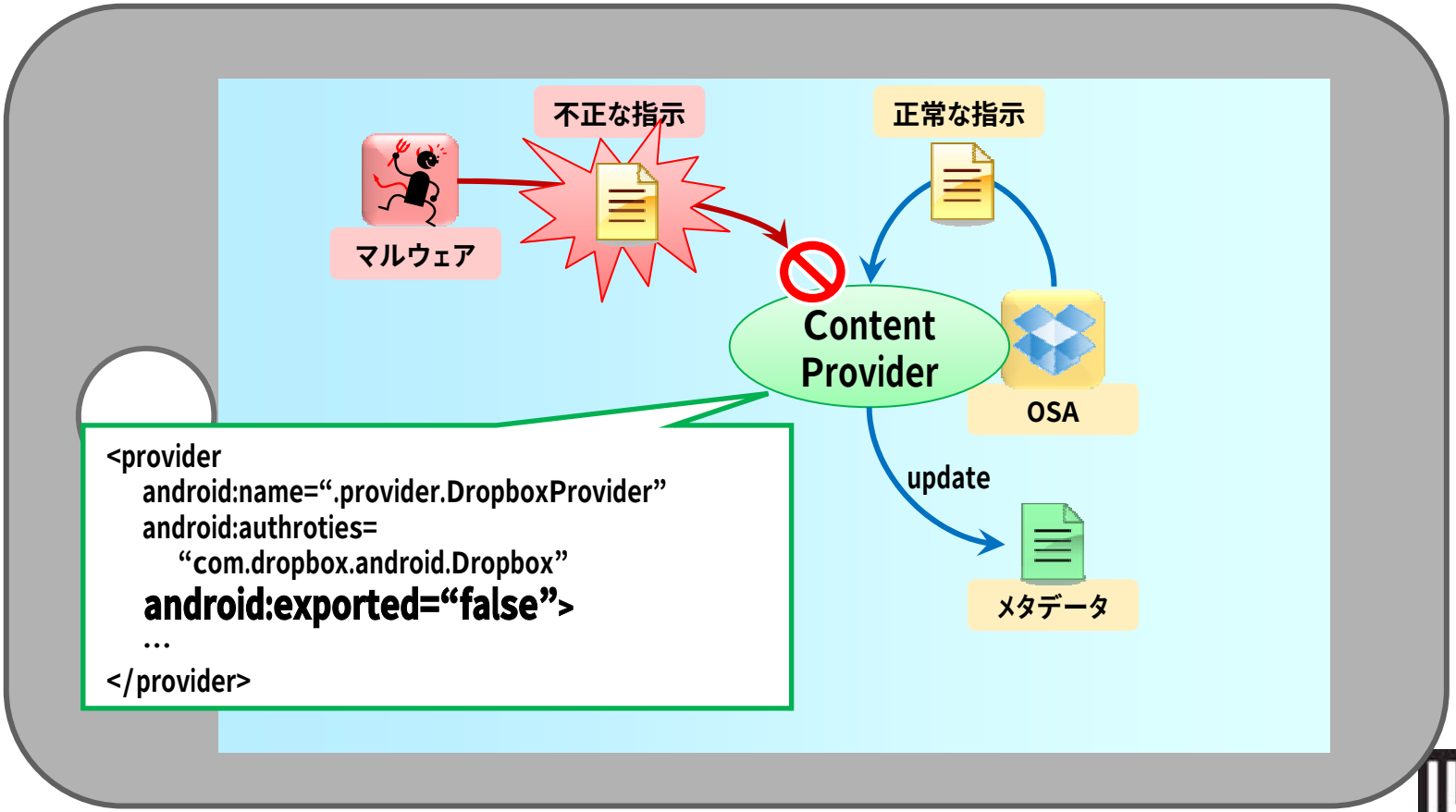
脆弱性の原因

同期を指示する情報(下図のメタデータで扱う情報)を
ContentProvider経由で他アプリから操作できる



対策

他アプリから利用すべきでないContentProviderは、
exported属性を”false”にして、非公開にする。



OSVDB#74648
某オンラインストレージアプリにおける
ContentProviderアクセス範囲の脆弱性

セキュアコーディングガイドを参照する

4.3.1.1. 非公開Content Provider を作る・利用する

ポイント(Content Provider を作る):

1. Android 2.2(API Level 8)以前では非公開 Content Provider を実装しない(できない)
2. exported="false"により、明示的に非公開設定する
3. 同一アプリ内からのリクエストであっても、パラメータの安全性を確認する
4. 利用元アプリは同一アプリであるから、センシティブな情報を返送してよい

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.jssec.android.provider.privateprovider"
    android:versionCode="1"
    android:versionName="1.0" >

    <!-- ★ポイント1★ Android 2.2 (API Level 8) 以前では非公開 Content Provider を実装しない（できない） -->
    <uses-sdk android:minSdkVersion="9" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".PrivateUserActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- ★ポイント2★ exported="false"により、明示的に非公開設定する -->
        <provider
            android:name=".PrivateProvider"
            android:authorities="org.jssec.android.provider.privateprovider"
            android:exported="false" />
    </application>
</manifest>
```

**AndroidManifest.xmlで
exported属性をfalseに指定し
非公開にする**

JAPAN SMARTPHONE SECURITY ASSOCIATION

セキュアコーディングガイド 2012年11月1日版

トピック紹介

JAPAN SMARTPHONE SECURITY ASSOCIATION

What's New!

日付	改訂内容
2012-06-01	• 初版
2012-11-01	• 下記の構成・内容を見直し拡充致しました • 4.1 Activity を作る・利用する • 4.2 Broadcast を受信する・送信する • 4.3 Content Provider を作る・利用する • 4.4 Service を作る・利用する • 5.2 Permission と Protection Level
	• 下記の新しい記事を追加致しました • 2.5 サンプルコードの Eclipse への取り込み手順 • 3.1 Android アプリのセキュリティ ← 解説します • 4.7 Browsable Intent を利用する • 5.3 Account Manager で独自アカウントを追加する • 6.1 Clipboard から情報漏洩する危険性
• 新版の公開にあたり、皆様から頂いたご意見・コメントを元に本ガイドの内容を更新しました	

Androidアプリのセキュリティ

セキュリティの考え方

1.資産

3.対策

2.脅威

JAPAN SMARTPHONE SECURITY ASSOCIATION

1.資産

資産

- システムやアプリにおいて守るべき対象のことを資産と呼ぶ
- 守るべき対象には情報と機能の2つがあり、それぞれ情報資産と機能資産と呼ぶ

情報資産	• 許可された人・アプリだけが参照や変更できる情報 • その他の人・アプリには参照や変更ができてはならない情報
機能資産	• 許可された人・アプリだけが利用できる機能 • その他の人・アプリには利用できてはならない機能

情報資産

Android OSが管理する情報資産
端末の電話番号
電話帳
通話履歴（受発信の日時や番号）
IMEI（携帯電話の端末ID）
IMSI（回線契約者ID）
設定情報（WiFi設定…）
アカウント情報（Googleアカウント…）
メディアデータ（写真、動画、音楽、録音…）
…

アプリ	アプリが管理する情報資産
Eメール	メアド、送受信メール本文、添付…
SMS	送受信SMSメッセージ本文…
Webブラウザ	ブックマーク、閲覧履歴、クッキー…
カレンダー	予定、ToDo、イベント…
家計簿	残高、買い物履歴…
Facebook	アカウント、コンテンツ…
Twitter	アカウント、コンテンツ…
mixi	アカウント、コンテンツ…
…	

- 各々の情報資産には誰がアクセスできて、誰がアクセスできてはならないのかといったセキュリティ要件がある
- アプリはスマホ内の様々な情報資産にアクセスできるが、セキュリティ要件を逸脱しない設計・実装が求められる

機能資産

Android OSが提供する機能資産
ネットワーク通信機能
カメラ撮影機能
電話を掛ける機能
携帯電話状態の読み取り機能
ログデータの読み取り機能
GPS等で現在位置を得る機能
SMSメッセージを送受信する機能
SDカード書き込み機能…

アプリ間連携のために、アプリが他のアプリに提供する機能資産

写真撮影アプリで写真を撮影し、写真装飾アプリで写真を装飾し、装飾した写真をFacebookに投稿するアプリ間連携の一例。

- 各機能資産にも誰がアクセスできて、誰がアクセスできてはならないのかといったセキュリティ要件がある
- アプリはスマホ内の様々な機能資産にアクセスできるが、セキュリティ要件を逸脱しない設計・実装が求められる


2.脅威

脅威

- セキュリティ要求に反する資産へのアクセス行為（攻撃行為）を脅威と呼ぶ

	セキュリティ要求	脅威
情報資産	• 許可された人・アプリだけが参照や変更できる • その他の人・アプリには参照や変更ができてはならない	• 許可されていない人・アプリが不正に参照や変更する行為 • 許可されていない人・アプリが許可された人・アプリの正当な参照や変更を妨害する行為
機能資産	• 許可された人・アプリだけが利用できる機能 • その他の人・アプリには利用できてはならない機能	• 許可されていない人・アプリが不正に利用する行為 • 許可されていない人・アプリが許可された人・アプリの正当な利用を妨害する行為

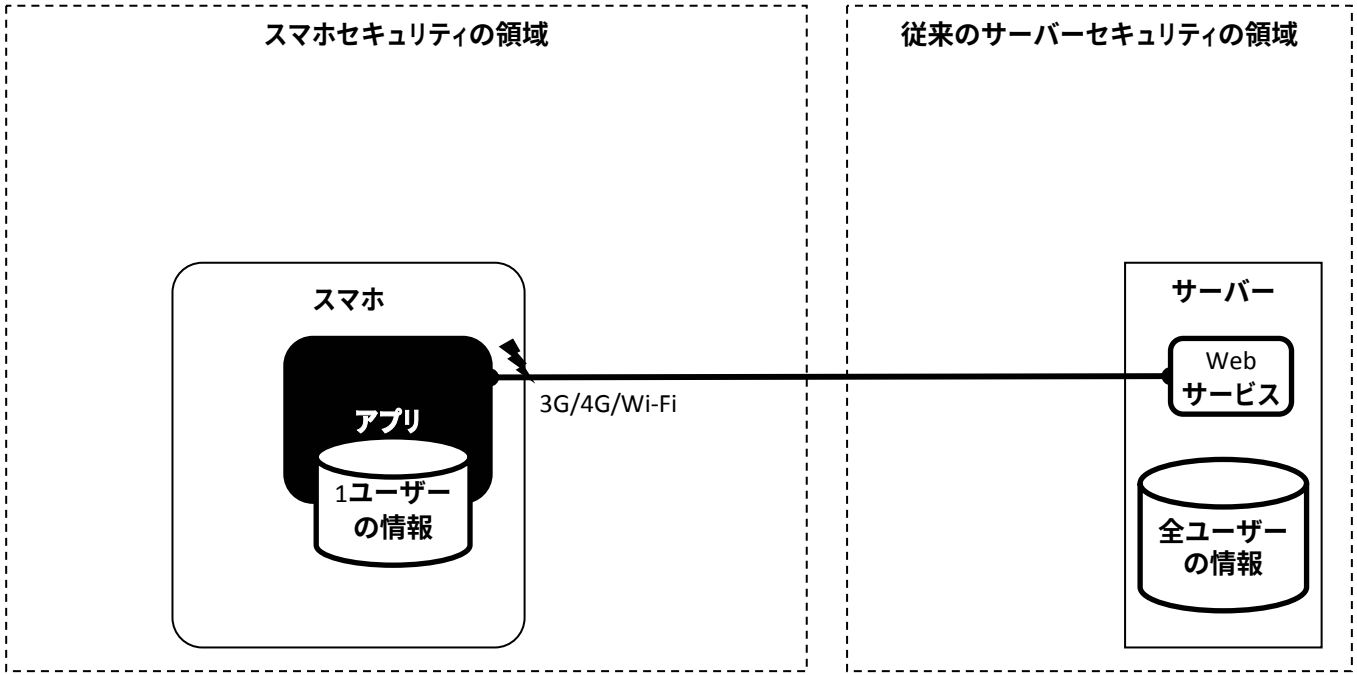
はい、あくびをどうぞ！

- 理屈っぽいと眠くなりますよね

- この後は図解で脅威を紹介していきます

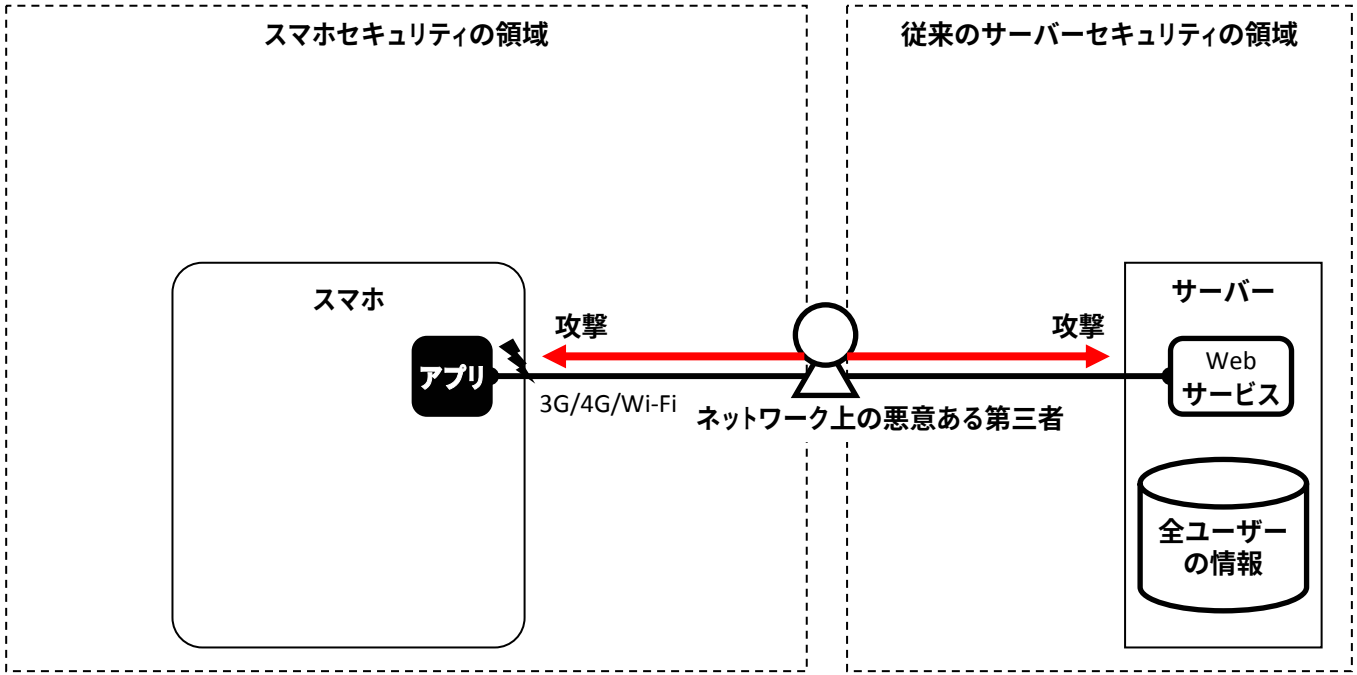
JAPAN SMARTPHONE SECURITY ASSOCIATION

一般的なスマホアプリ構成



JAPAN SMARTPHONE SECURITY ASSOCIATION

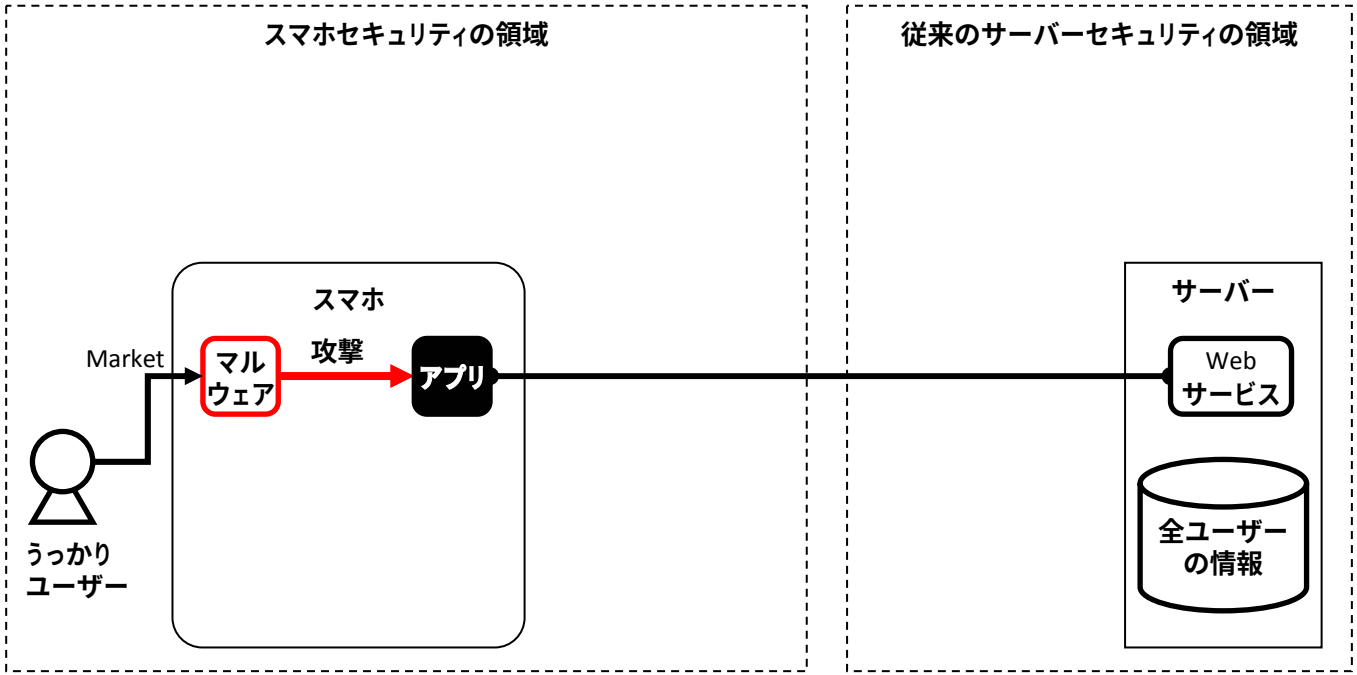
ネットワーク上の悪意ある第三者



- **通信中のデータを盗聴・改竄**
- **(スマホに対して) サーバーに成りすましたり、**
  (サーバーに対して) アプリやユーザーに成りすまし

JAPAN SMARTPHONE SECURITY ASSOCIATION

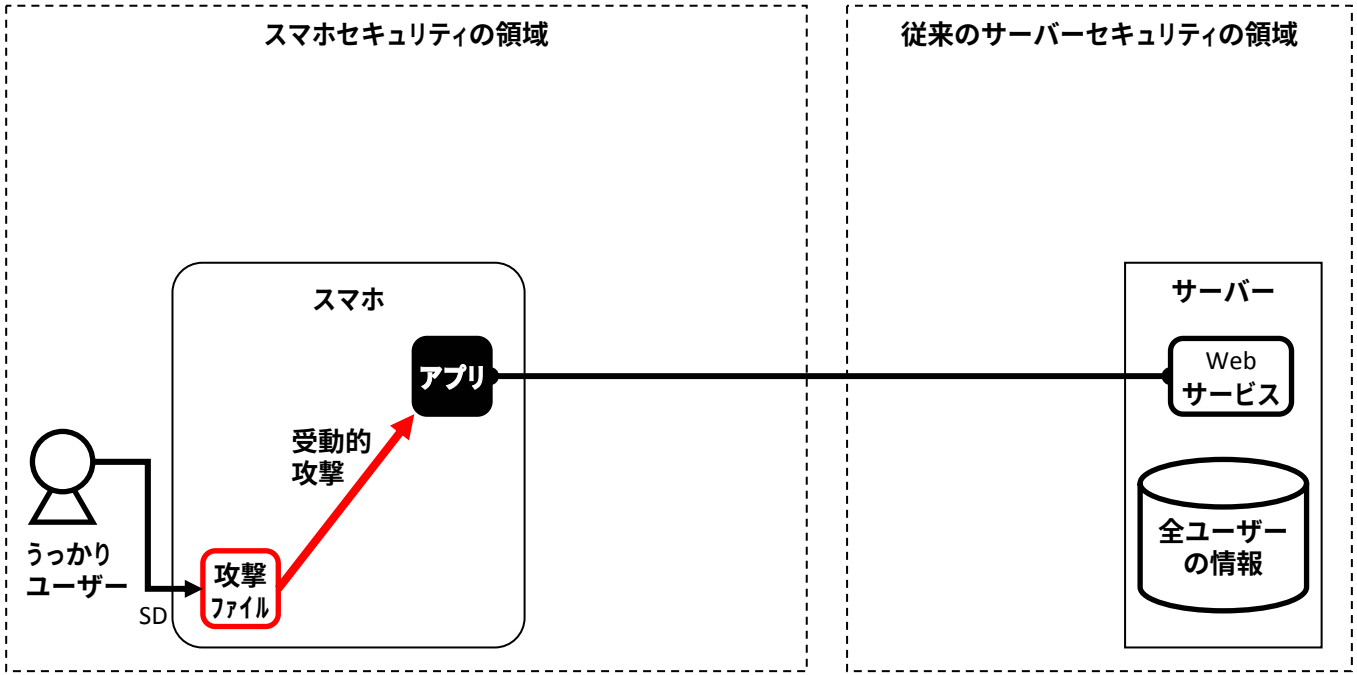
ユーザーが導入したマルウェア



- アプリが管理している情報を盗聴・改竄・破壊
- アプリの機能を勝手に呼び出し

JAPAN SMARTPHONE SECURITY ASSOCIATION

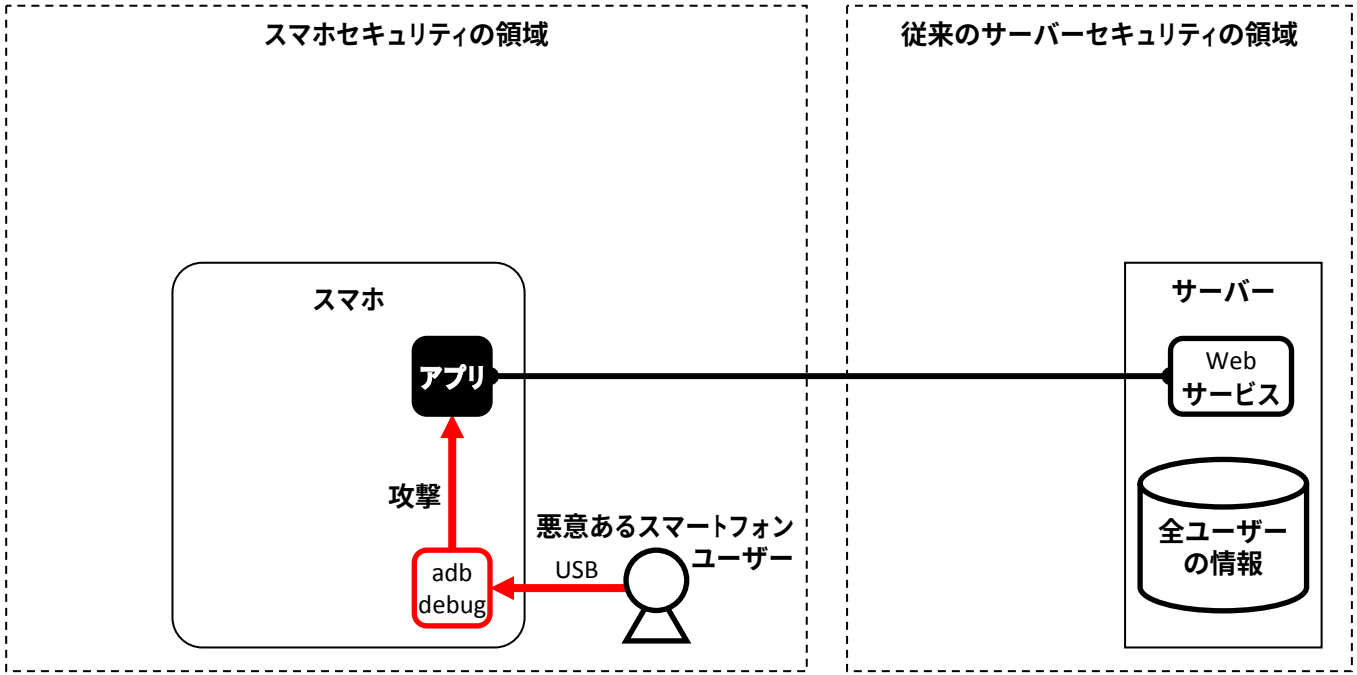
脆弱性を攻撃する悪意あるファイル
(画像や動画ファイルなど)



- アプリが攻撃ファイルを開くと、アプリの脆弱性が突かれ、アプリが管理する情報やアプリの機能が悪用される

JAPAN SMARTPHONE SECURITY ASSOCIATION

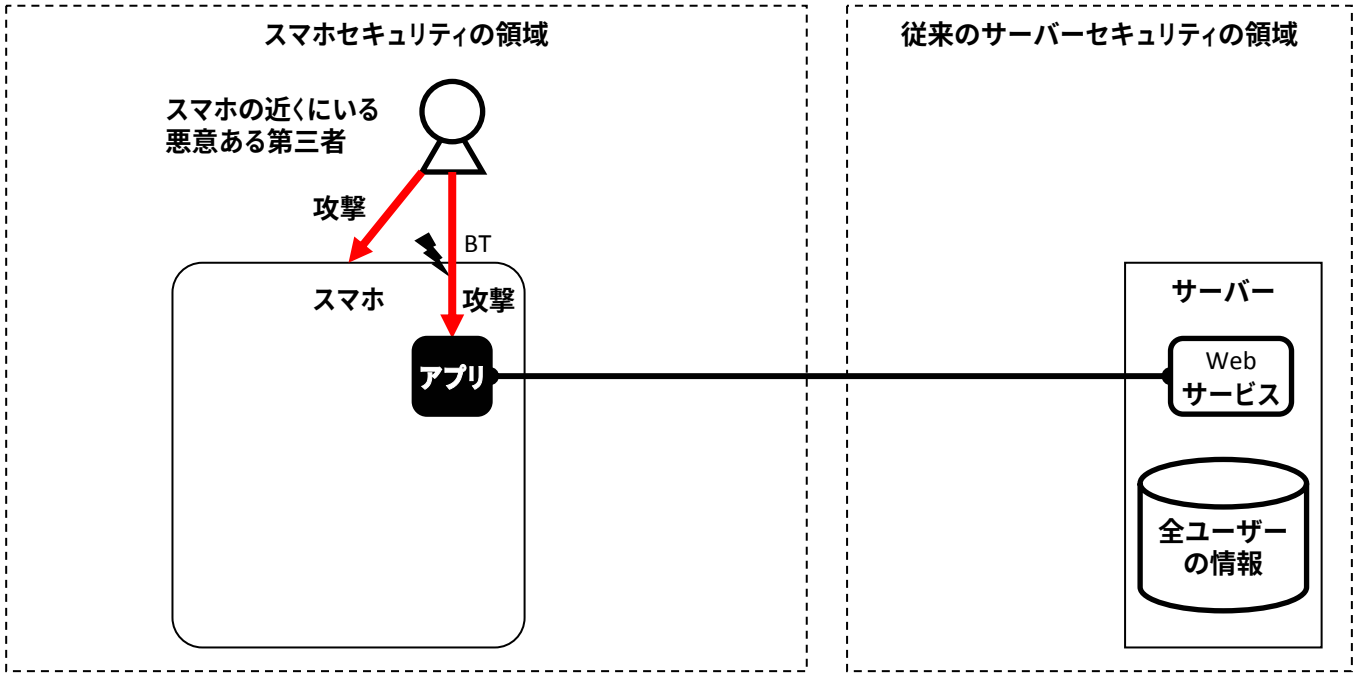
悪意あるスマホユーザー



- デバッグ機能を悪用し、アプリの中にあるメーカーの機密情報を読み出したり、アプリの機能を悪用したり

JAPAN SMARTPHONE SECURITY ASSOCIATION

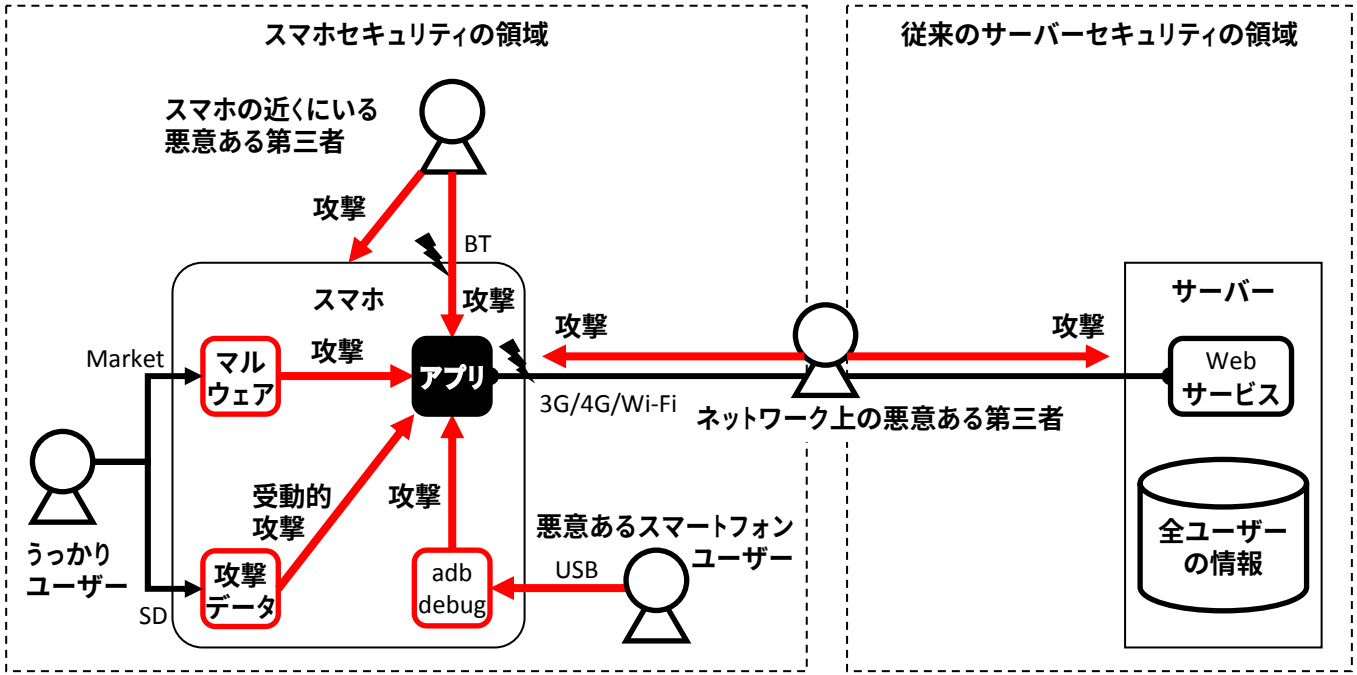
スマホの近くにいる悪意ある第三者



- ショルダーハック（画面覗き込み）
- Bluetooth等の近距離無線経由でアプリを攻撃
- スマホ端末自体を窃盗、破壊

JAPAN SMARTPHONE SECURITY ASSOCIATION

アプリはあちこちから狙われる



- これがすべての脅威ではない
- アプリをとりまくさまざまな攻撃を想定して、アプリをつくらなければならない

JAPAN SMARTPHONE SECURITY ASSOCIATION

3.対策

対策

- 脅威から資産を守る保護施策をセキュリティ対策と呼ぶ

- どのように守るか？
  - 様々な技術的なテクニックがある
  - セキュアコーディングガイドの各記事を参照

- どれくらい守るか？

⇦ 考え方を解説します

一番よく聞かれる質問

- **質問：**
  - Androidアプリでセキュリティ確保は大変。rootが取られたら守りきれないですよね？
- **回答：**
  - 確かに、Android OSのセキュリティモデルで守ることはできなくなりますが、難読化や暗号化、ハードウェア支援等を組み合わせて相当頑張ったら、守れないこともないです。

ところで・・・

  - あなたのアプリが扱う資産は端末のrootを奪取してまで狙われるほど攻撃者に価値あるものですか？
  - 端末のrootが奪取され、あなたのアプリの資産が狙われたとして、被害は許容できないくらい大きくなりますか？

Android OSの限界ばかりが気になっていて、自分のアプリのことをよく把握していないというケースがよくあります ☺

どれくらい守るか？

- Androidアプリには多くの脅威が存在し、そのすべてに対応するには限界がある
- 資産の重要度に応じて妥当な保護施策を決定するのが現実的である

次の手順が一般的

1. アプリが扱う各々の資産を重要度で分類
2. 重要度に応じて保護施策のレベルを決定

各々の資産を重要度で分類

資産分類	資産の重要度	例えば…
高位	資産が被害にあった場合、組織または個人の活動に致命的または壊滅的な影響をあたえるもの	資産が被害にあった場合、当該組織が事業を継続できなくなるレベル
中位	資産が被害にあった場合、組織または個人の活動に重大な影響をあたえるもの	資産が被害にあった場合、当該組織の利益が悪化し、事業に影響を及ぼすレベル
低位	資産が被害にあった場合、組織または個人の活動に限定的な影響をあたえるもの	資産が被害にあった場合、当該組織の利益に影響を与え得るが、他の要素により利益の補てんが可能なレベル

重要度に応じて保護施策を決定

資産分類	資産の重要度	保護施策のレベル
高位	壊滅的な影響	• Android OSのセキュリティモデルを破る、root権限を奪取した状態からの攻撃やAPKのdex部分を改造するといった、**高度な攻撃に対しても保護**する • UX等の他要素よりもセキュリティ確保を優先する
中位	重大な影響	• Android OSのセキュリティモデルを活用し、その範囲内で保護する • UX等の他要素よりも**セキュリティ確保を優先**する
低位	限定的な影響	• Android OSのセキュリティモデルを活用し、その範囲内で保護する • UX等の他要素とセキュリティを比較し、**他要素を優先しても良い**

セキュアコーディングガイドのスコープ外

専門家に相談してください

セキュアコーディングガイドの対象範囲

JAPAN SMARTPHONE SECURITY ASSOCIATION

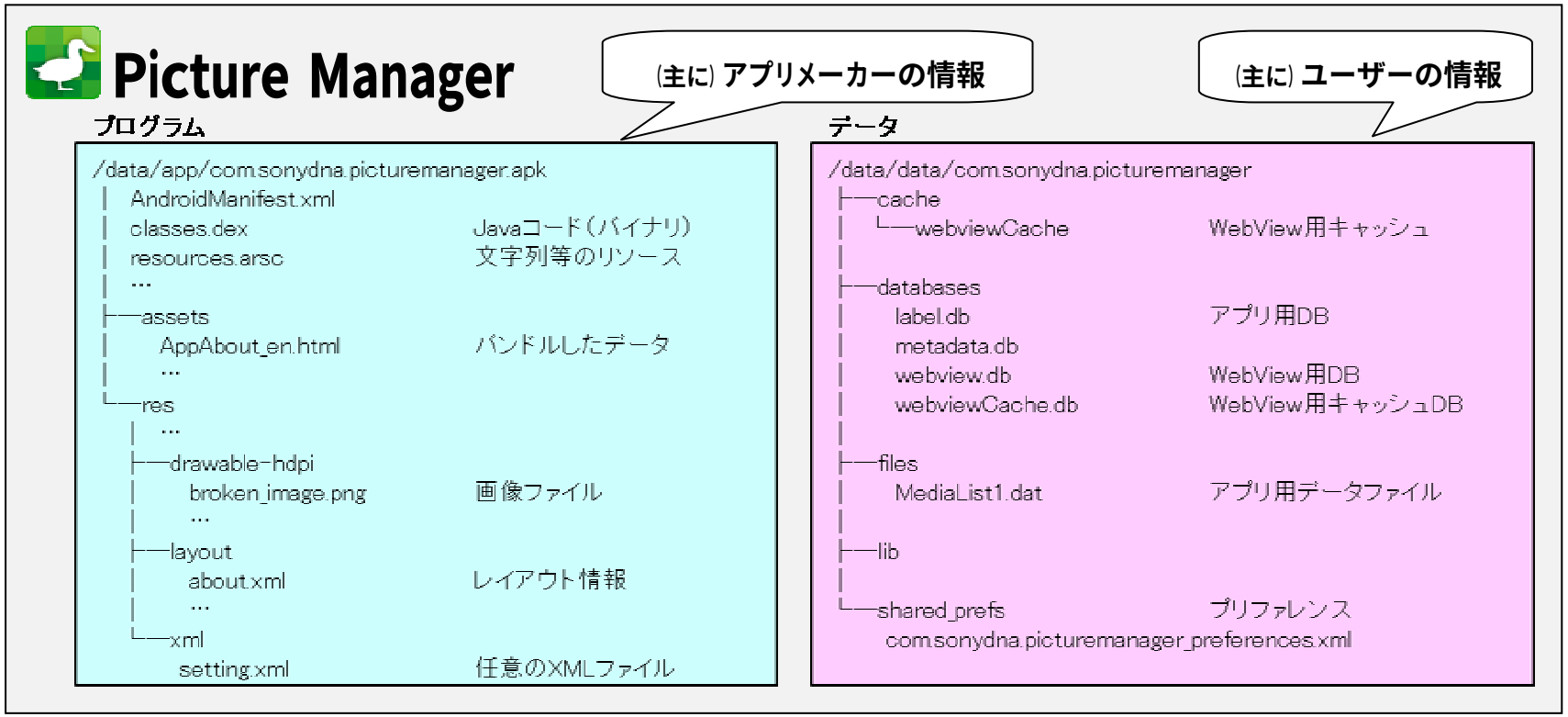
まとめ

1. Androidアプリ脆弱性の動向
   - 今年に入ってからAndroidアプリの脆弱性報告が急増している状況を紹介
2. Androidアプリ脆弱性事例とガイド
   - ガイドを読んでいたらよかったかもという脆弱性事例を紹介
3. ガイド第2版のトピック紹介
   - どれくらい守るか？に対し、資産の重要度に応じて妥当な保護施策を決定する現実的な方法を紹介

Backup

JAPAN SMARTPHONE SECURITY ASSOCIATION

アプリの中の情報資産



- アプリメーカーの情報の中には、ユーザーには見せたくない情報、勝手に利用されたくない情報もある

電話帳 (Contacts) の 1件のエントリに含まれる情報資産

情報	内容
名前	表示名、名前、苗字、ミドルネーム…
電話番号	自宅、携帯電話、仕事、FAX、MMS…
Eメールアドレス	自宅、仕事、携帯電話…
プロフィール画像	サムネール画像、大きな画像…
インスタントメッセンジャー	AIM、MSN、Yahoo、Skype、QQ、Google Talk、ICQ、Jabber、Netmeeting…
ニックネーム	略称、イニシャル、旧姓、別名…
住所	国、郵便番号、地域、地方、町、通り…
グループ	お気に入り、家族、友達、同僚…
ウェブサイト	ブログ、プロフィールサイト、ホームページ、FTPサーバー、自宅、会社…
イベント	誕生日、記念日、その他…
関係する人物	配偶者、子供、父、母、マネージャー、助手、同棲関係、パートナー…
…	

- スマホのユーザーに関する情報だけではなく、そのユーザーの知人、友人など、1ホップ先の人々の情報もスマホには含まれている