せきゆりてい侍

事例から学ぶAndroid セキュリティの傾向と対策

株式会社OpenTone
金融Solution事業部
Bee-Team

BeeTeam とは

- 玉侍、根侍、野侍、菱侍、川侍、大侍の6人の侍（ギーグ）衆
- おぷとんらぼにコラムを掲載中

  http://labs.opentone.co.jp/

自己紹介

- 名前：大羽久知（おおばひさし）　大侍
- 年齢：２０歳と２４０ヶ月
- 将来は憲兵団に入って内地勤務が夢
- 連絡先：hisashi.ohba@opentone.co.jp

Androidの情勢 - 成長するAndroid市場

- スマートフォン市場、特にAndroidユーザ数が急成長
- アプリダウンロード数

Androidの情勢 - 脅威

- 不正アプリ数が急増
  - PCの約4倍の速度で増加

出典：
トレンドマイクロ「2012年　世界の脅威動向レポート　セキュリティ・ラウンドアップ」

- モバイルマルウェアの　　Androidが標的

出典：
F-Secure「Mobile Threat Report Q4 2012」

本プレゼンテーションの流れ

- 前半
  - 情報流出の脅威
  - 代表的なセキュリティインシデント（マルウェア）
  - 利用者側の対策
  - 代表的なセキュリティインシデント（脆弱性）
  - 脆弱性に対する対策
  - セキュアプログラミングの必要性

- 後半
  - Androidにも潜む、脆弱性を産むコード
  - 攻撃とセキュアプログラミング
  - 脆弱性の原因

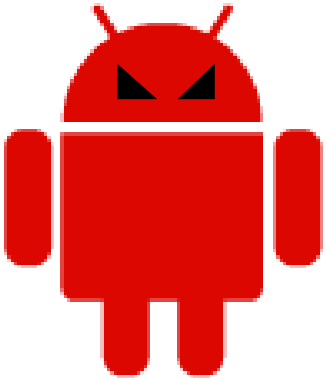
情報流出の脅威

- 200万個以上のアプリを解析
  - 29万3091個が「明らかに悪意のあるアプリ」

「高リスク」に分類	51%
GooglePlayで公開	23%
ユーザの情報を不適切に流出	22%

出典：
トレンドマイクロ「モバイル脅威の現状:Android向け不正アプリの23%がGoogle Play経由で提供」

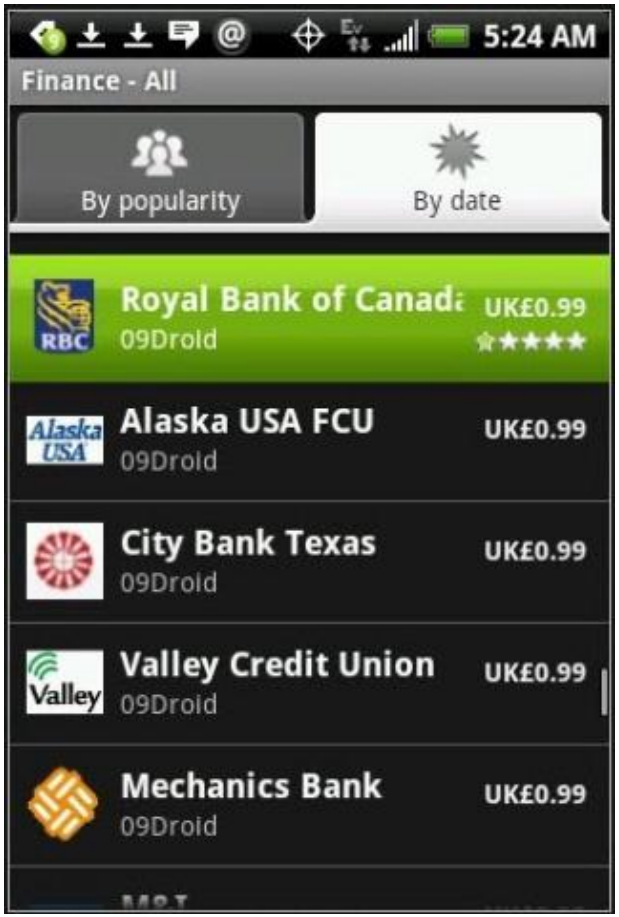
代表的なセキュリティインシデント（マルウェア）



- Droid09 (2010.1)
- Geinimi (2010.12)
- DroidDreamLight (2011.6)
- the Movie/動画まとめ (2012.4)

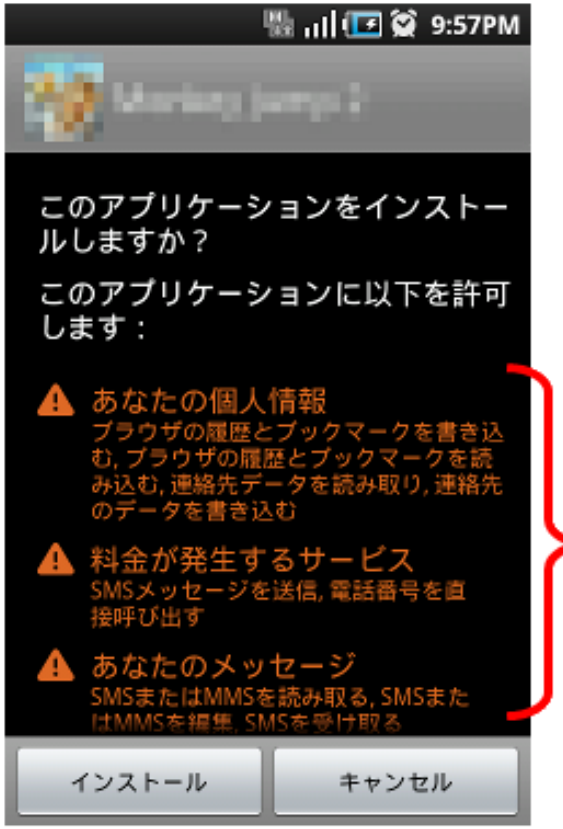
Droid09

- 2010.1発生
- オンラインバンキングアプリを偽装
- ユーザ名やパスワードなどが流出
- Android Market(当時)で公開



出典： F-Secure

Geinimi



- 2010.12発生
- Android初のボット型ウィルス
- 意図しない電話発信やメールの送受信、個人情報の漏えいなどが起こる

不審なアクセス許可が求められている

出典： IPA「Android OSを標的としたウイルスに関する注意喚起」

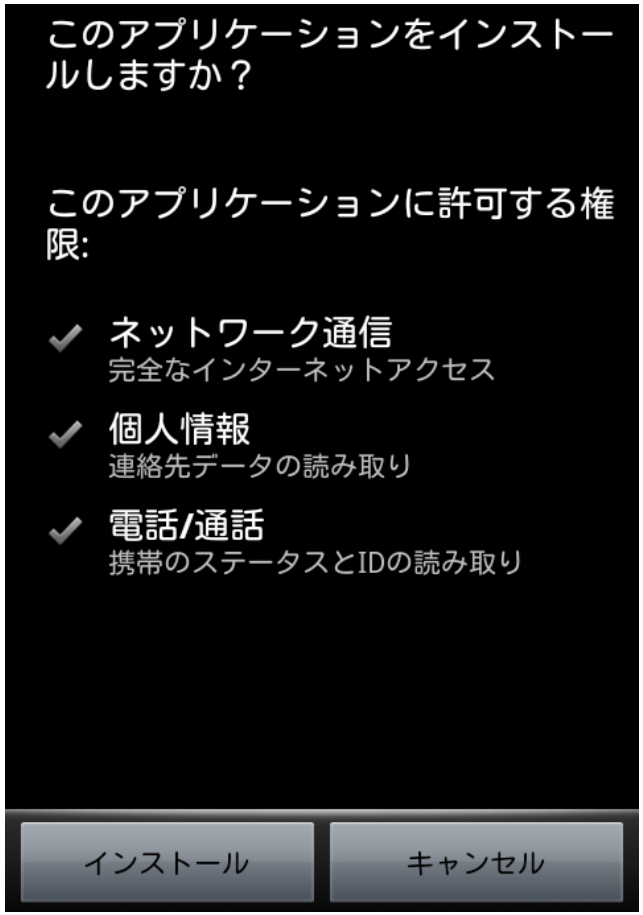
DroidDreamLight

- 2011.6発生
- トロイの木馬
- ユーザ端末から個人情報等が流出
- Google Playで公開

DroidDreamLightの感染フロー

出典： ITpro「Androidを狙う「DroidDreamLight」を知る」

the Movie/動画まとめ



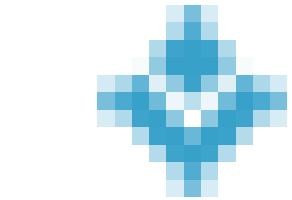
- 2012.4発生
- 「～the Movie」「～動画まとめ」アプリで、数万件～数百万件の個人情報が大量流出
- Google Playで公開

出典： シマンテック「日本の Android ユーザーから個人情報を盗み出す "The Movie" マルウェア」

利用者側の対策

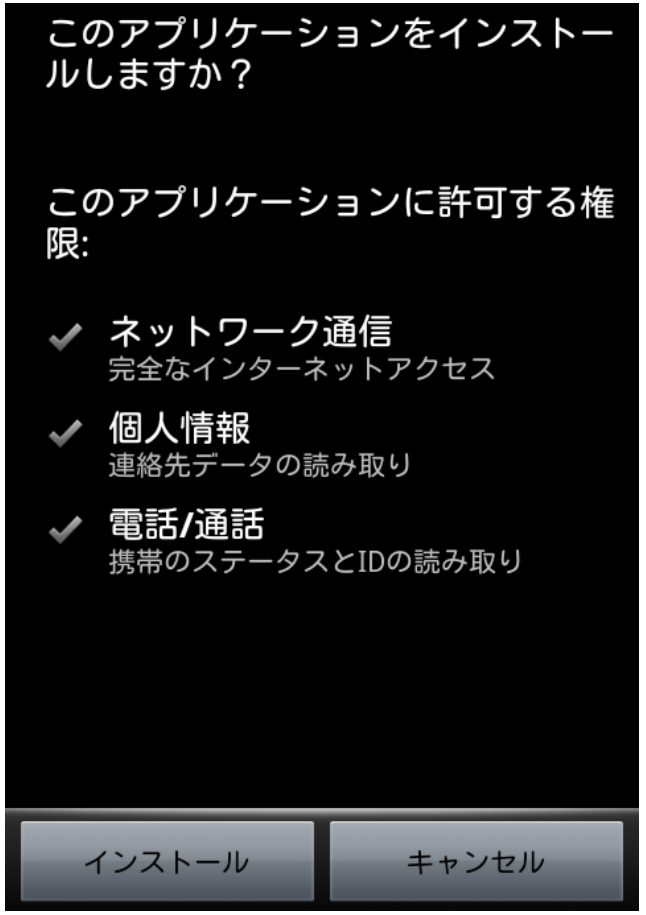
- 信頼できる公式アプリマーケットから入手する
  - 携帯キャリアの公式サイト
    - 審査が厳しい
  - Google Play
    - トップデベロッパー

Google Play チームが選んだトップ レベルの Google Play デベロッパー



利用者側の対策

- 「パーミッション」の確認
  - Android特有のセキュリティ機構
  - ユーザーがインストールの可否を判断
    (アプリが必要とする権限を開発者が明示)



出典：@IT「Security & Trust - Androidセキュリティの今、これから」

利用者側の対策

マルウェアによく使われるパーミッション

ラベル名	パーミッション名
SMSの送信	SEND_SMS
SMSの受信	RECEIVE_SMS
連絡先データの読み取り	READ_CONTACTS
起動時に自動的に開始	RECEIVE_BOOT_COMPLETED
完全なインターネットアクセス	INTERNET
携帯のステータスとIDの読み取り	READ_PHONE_STATE
おおよその位置情報（ネットワーク基地局）	ACCESS_COARSE_LOCATION
精細な位置情報（GPS）	ACCESS_FINE_LOCATION

出典：@IT「Security & Trust - Androidセキュリティの今、これから」
@IT「 Androidを取り巻く脅威――ユーザーにできることは？」

利用者側の対策

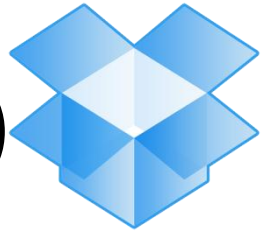
- セキュリティアプリの導入
  - 有料、無料問わず多くのベンダーからリリース
  - ただし、PC向けほど効果に期待はできない
    (一定の効果はあるが、機能に限界があると理解した上で利用)
    - メールの添付ファイルに用心
    - 怪しいサイトには行かない
    - 「パーミッション」を確認する

出典：ITmedia「ホント？　PC並みには期待できないAndroidのウイルス対策アプリ」

代表的なセキュリティインシデント（脆弱性）

- Skype (2011.4)

- Dropbox (2011.8)

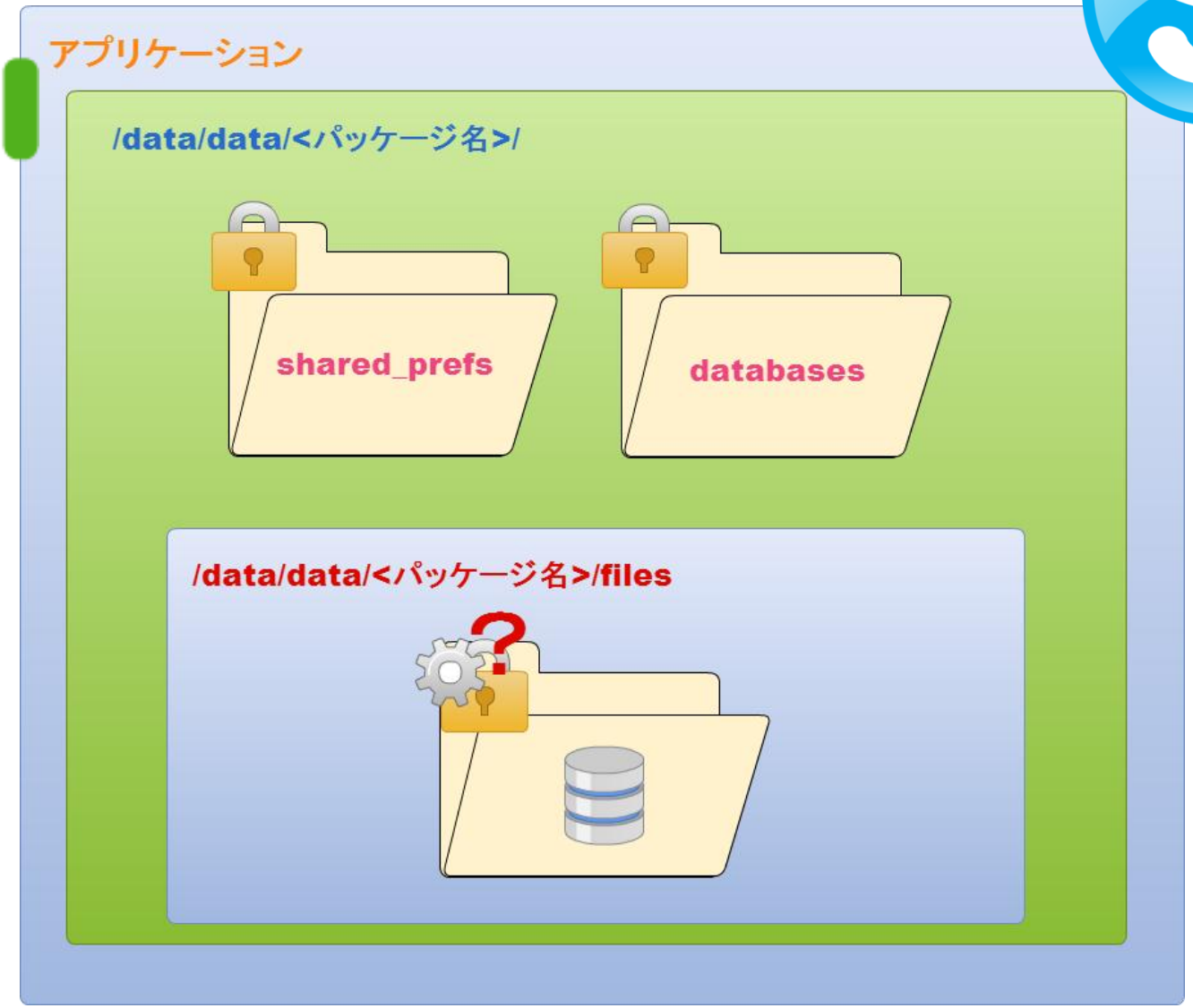


Skypeの脆弱性

- 2011.4発生
- 情報流出の恐れ
  - キャッシュされたプロファイル、インスタントメッセージ
- 原因
  - 独自の場所にデータを保存
  - アクセス許可が未設定

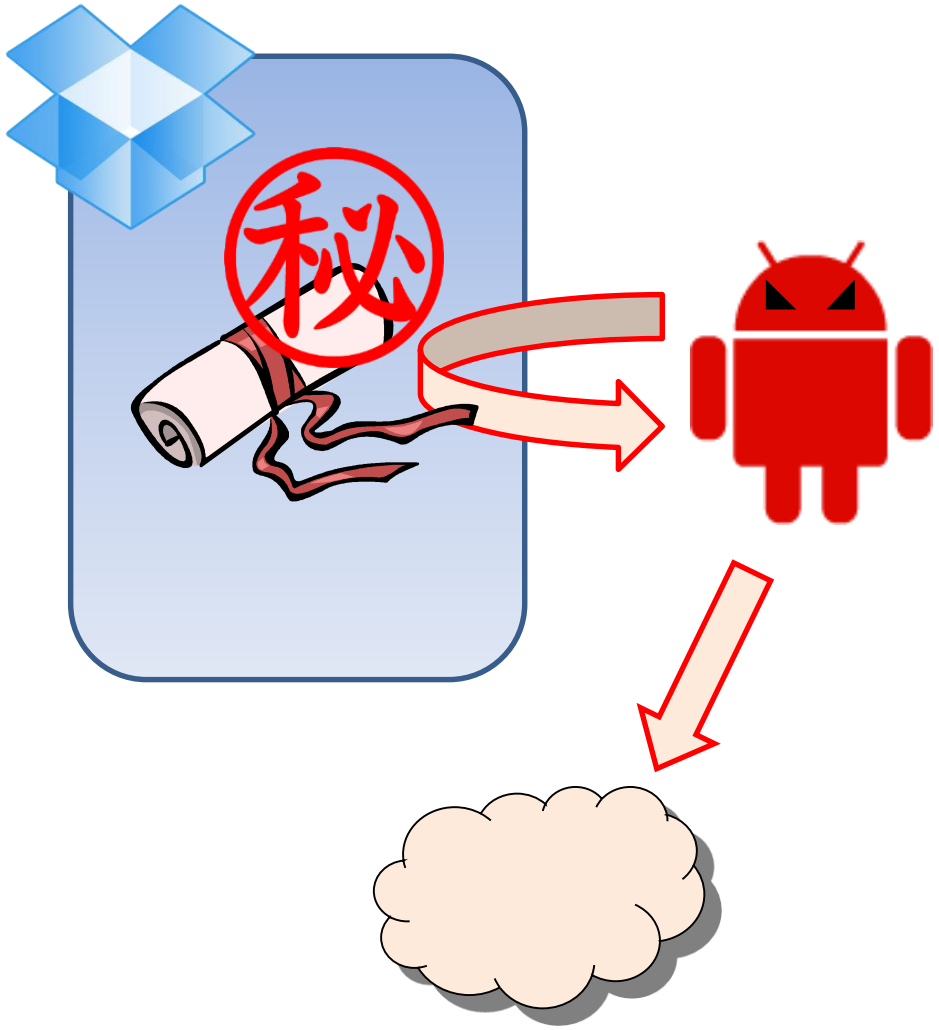
出典：
Think IT「Android Security　安全なアプリケーションを作成するために」

Skypeの脆弱性 (図解)



Dropboxの脆弱性

- 2011.8発生
- Dropboxのデータを別アプリから自由に読み書き
  - 意図せず個人情報を流出させてしまう可能性
- ContentProviderのアクセス制限が未設定



ContentProvider

- アプリケーション間でデータを共有するための仕組みの一つ
  - 通常、アプリケーションで保存されるデータは、他のアプリケーションからアクセスできない
- アクセス制限設定が出来る

ContentProviderのアクセス制限設定

- 公開

```
<provider android:name="xxx"
        android:authorities="xxx" />
```

- 非公開

```
<provider android:name="xxx"
        android:authorities="xxx"
        android:exported="false" />
```

セキュアプログラミングの必要性

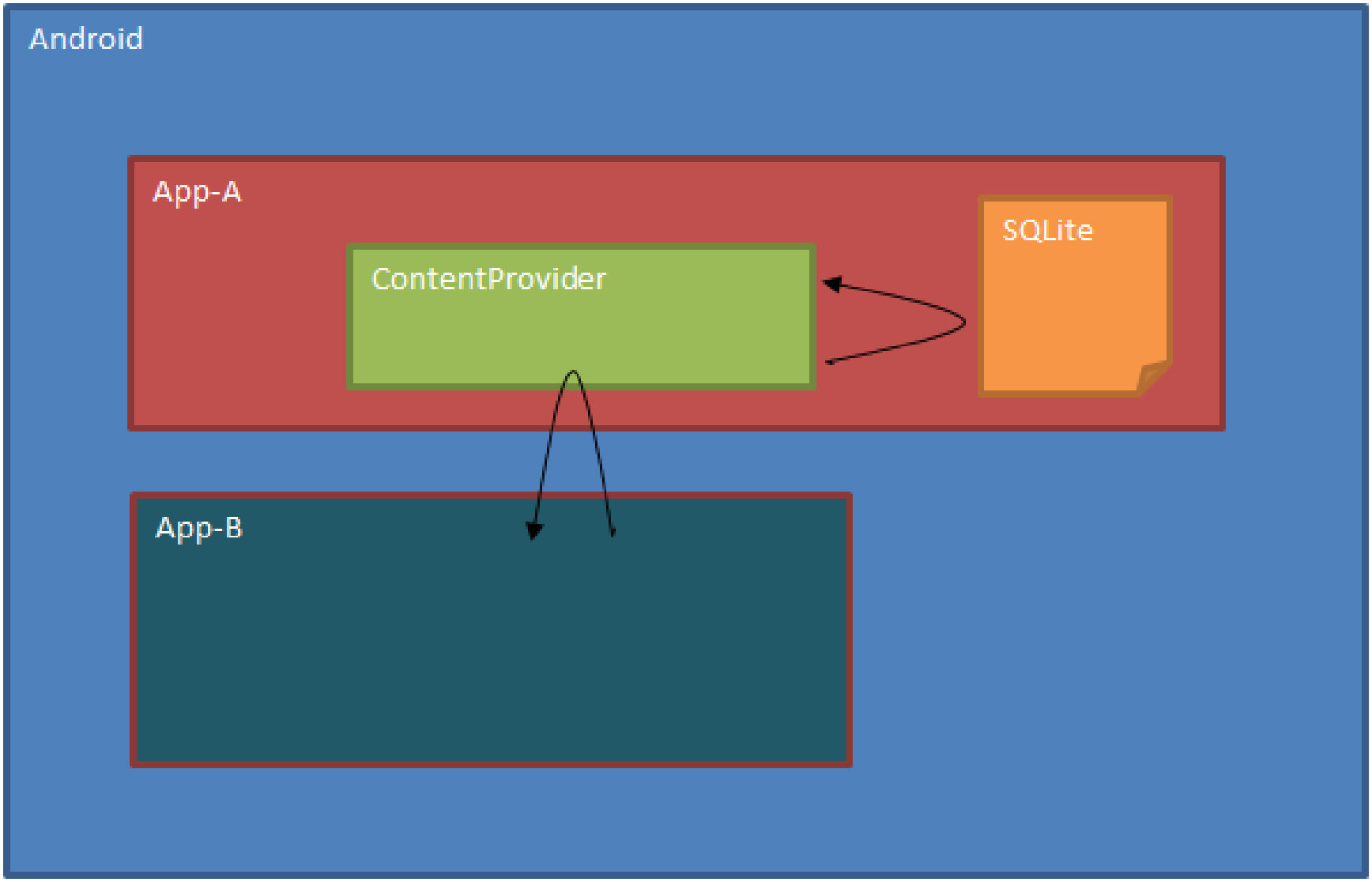
- Androidアプリは比較的容易に作成できる
- セキュアプログラミングの知識は重要！
- 意図せずセキュリティ被害の加害者に！！

本プレゼンテーションの流れ

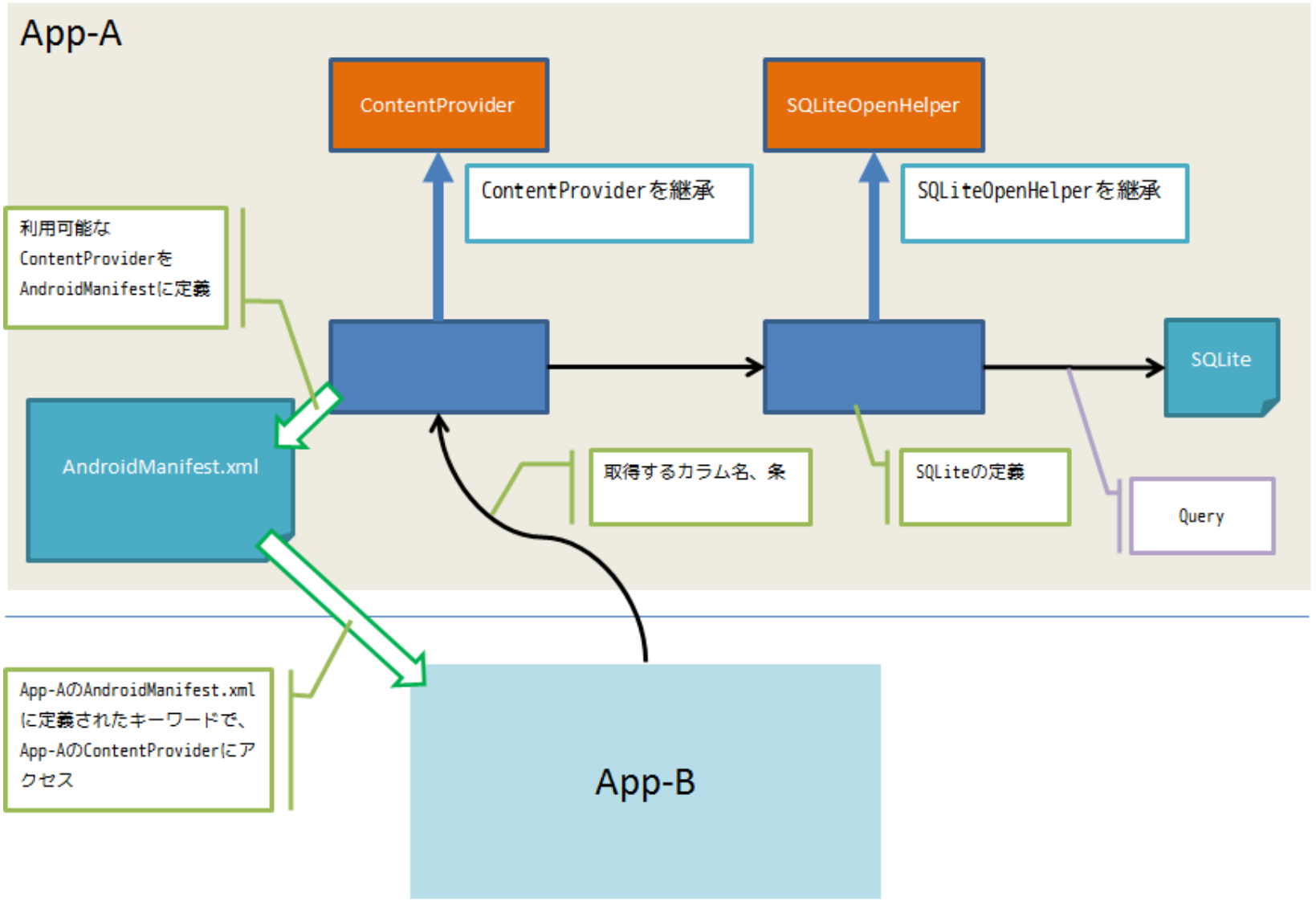
- 前半
  - 情報流出の脅威
  - 代表的なセキュリティインシデント（マルウェア）
  - 利用者側の対策
  - 代表的なセキュリティインシデント（脆弱性）
  - 脆弱性に対する対策
  - セキュアプログラミングの必要性

- 後半
  - *Android*にも潜む、脆弱性を産むコード
  - 攻撃とセキュアプログラミング
  - 脆弱性の原因

サンプルアプリ概要

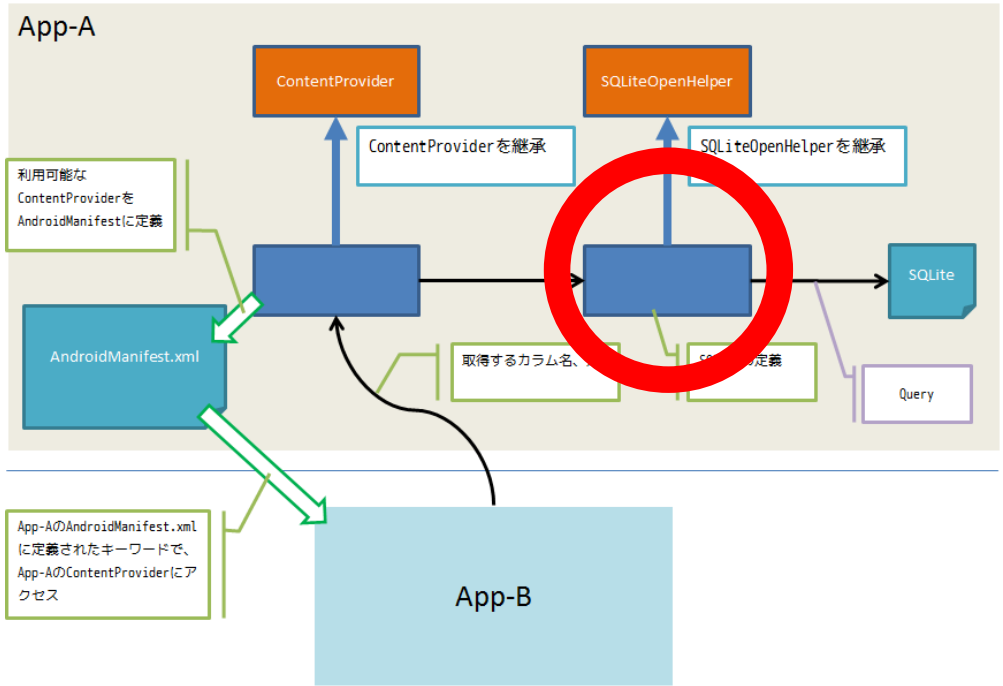


サンプルアプリ詳細


App-A
ContentProvider
SQLiteOpenHelper
ContentProviderを継承
SQLiteOpenHelperを継承
利用可能な
ContentProviderを
AndroidManifestに定義
SQLite
AndroidManifest.xml
取得するカラム名、条
SQLiteの定義
Query
App-AのAndroidManifest.xml
に定義されたキーワードで、
App-AのContentProviderにア
クセス
App-B

データベースを指定

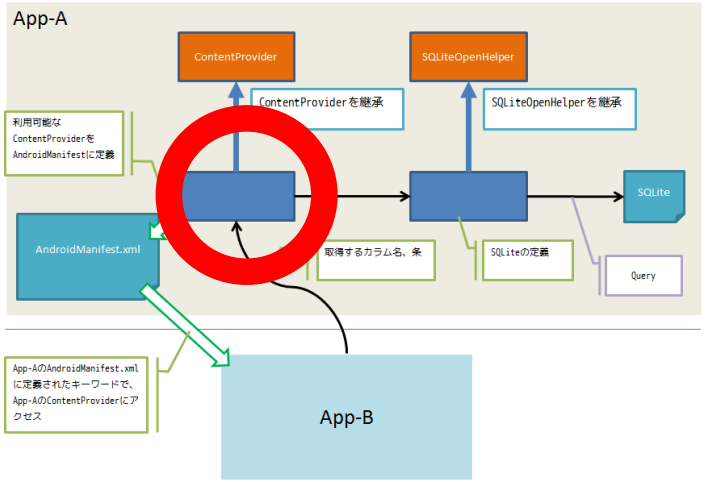
```
public class BeeDBHelper extends SQLiteOpenHelper {

    public BeeDBHelper(Context context) {
        super(context, "fsol-bee.db", null, 1);
    }
}
```



BeeDBHelper.java

```
public class BeeProvider extends ContentProvider {

    BeeDBHelper databaseHelper;

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {

        SQLiteDatabase db = databaseHelper.getReadableDatabase();
        SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
        qb.setTables("schedule_tbl");
        Cursor c = qb.query(db, projection, selection, selectionArgs,
                null, null, sortOrder);

        return c;
    }
```

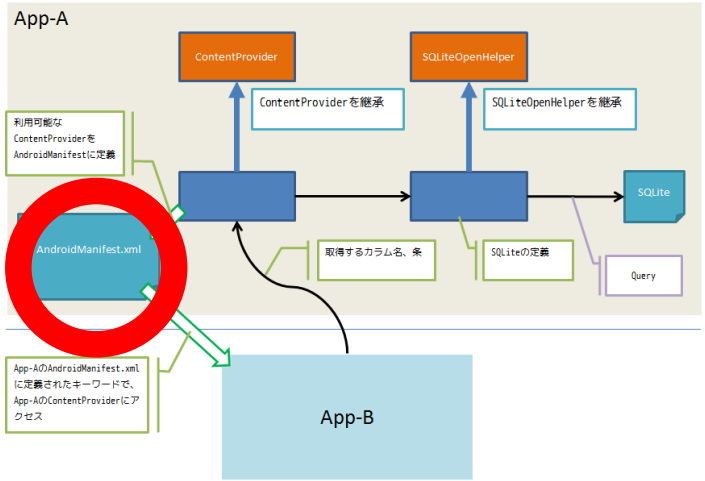
テーブルを指定



BeeProvider.java

ContentProviderの定義を追加

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="jp.co.opentone.fsol"
    android:versionCode="1"
    android:versionName="1.0" >
    :
        <provider
            android:name="BeeProvider"
            android:authorities="jp.co.opentone.fsol.beeprovider"
            android:exported="true" />
    :
</manifest>
```



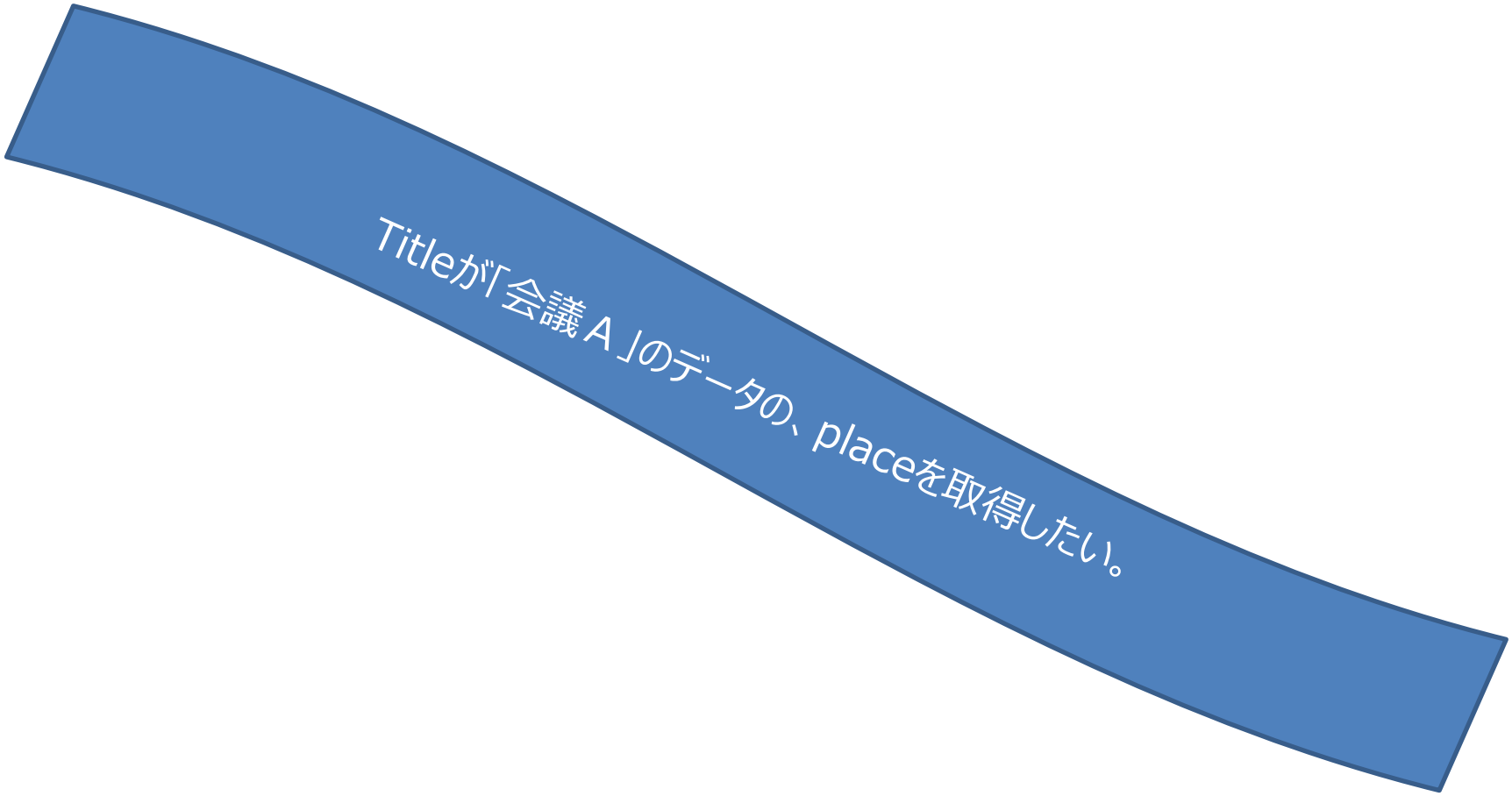
AndroidManifest.xml

contentに「会議」が含まれるデータの、title/content/dateを取得したい。

title	content	date	place
会議Ａ	超重要な会議	2013-04-29	会議室Ａ
会議Ｂ	定例会	2013-04-30	大会議室
会議Ｃ	グダグダな会議	2013-05-01	給湯室

```
String keyword = "会議";
Uri contentUri = Uri.parse(
        "content://jp.co.opentone.fsol.beeprovider");

try {
    Cursor cur = getContentResolver().query(contentUri,
            new String[] { "title", "content", "date" },
            "content LIKE ?", new String[] { "%" + keyword + "%" },
            null);
                :
```

title	content	date	place
会議Ａ	超重要な会議	2013-04-29	会議室Ａ
会議Ｂ	定例会	2013-04-30	大会議室
会議Ｃ	グダグダな会議	2013-05-01	給湯室



title	content	date	place
会議A	超重要な会議	2013-04-29	会議室A
会議B	定例会	2013-04-30	大会議室
会議C	グダグダな会議	2013-05-01	給湯室

```
String keyword = "会議A";
Uri contentUri = Uri.parse(
        "content://jp.co.opentone.fsol.beenormalprovider");

try {
    Cursor cur = getContentResolver().query(contentUri,
            new String[] { "place" },
            "title = ? ", new String[] { keyword }, null);
                :
```

title	content	date	place
会議A	超重要な会議	2013-04-29	会議室A
会議B	定例会	2013-04-30	大会議室
会議C	グダグダな会議	2013-05-01	給湯室

```
public class BeeProvider extends ContentProvider {

    BeeDBHelper databaseHelper;

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {

        SQLiteDatabase db = databaseHelper.getReadableDatabase();
        SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
        qb.setTables("schedule_tbl");
        projection = new String[] { "title", "content", "date" };
        selection = "content LIKE '%" + selectionArgs[0] + "%'";
        selectionArgs = null;
        Cursor c = qb.query(db, projection, selection, selectionArgs,
                null, null, null);

        return c;
    }
```

取得できるカラム
および条件を
ContentProvide側で指定

contentに「会議」が含まれるデータの、title/content/dateを取得したい。

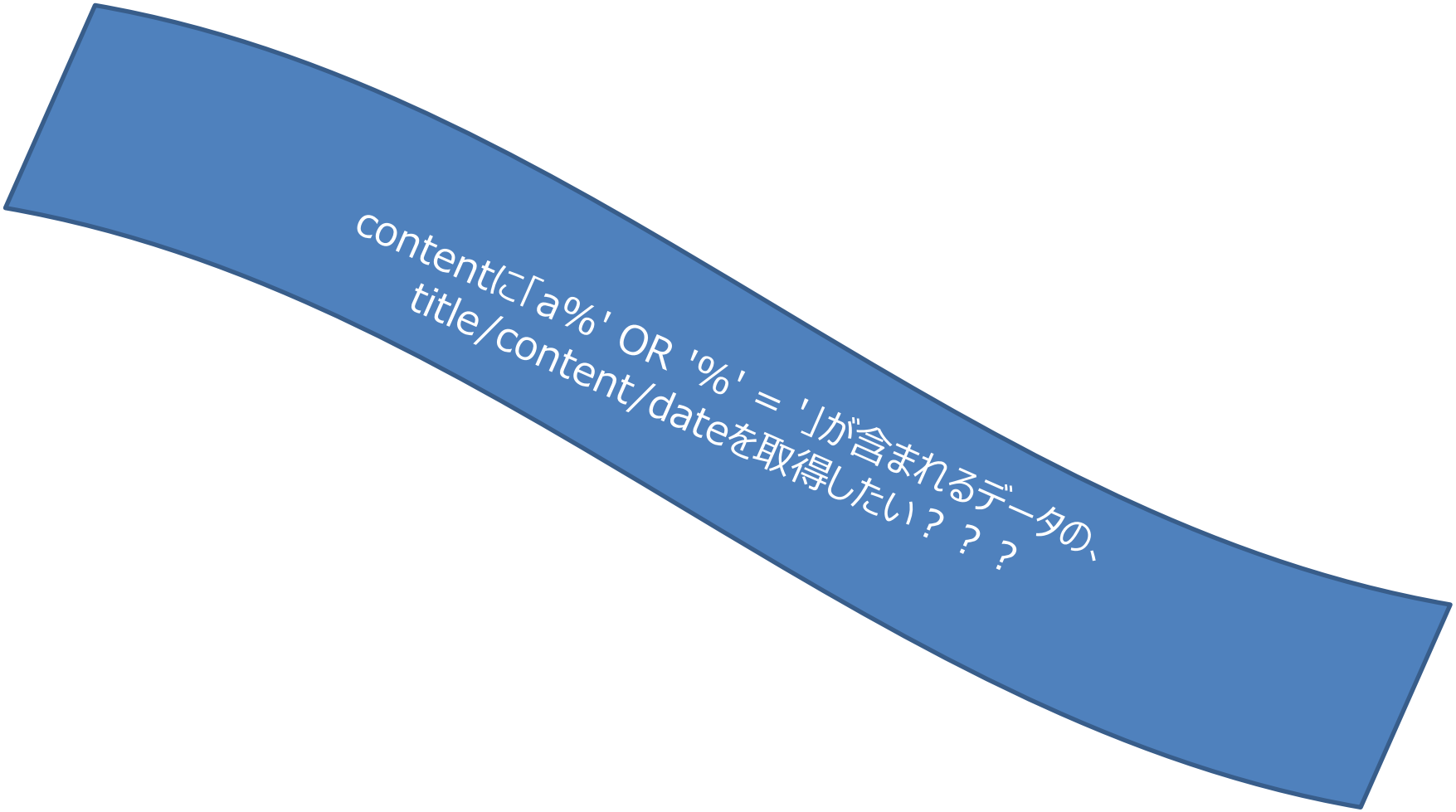
title	content	date	place
会議Ａ	超重要な会議	2013-04-29	会議室Ａ
会議Ｂ	定例会	2013-04-30	大会議室
会議Ｃ	グダグダな会議	2013-05-01	給湯室

```
String keyword = "会議";
Uri contentUri = Uri.parse(
        "content://jp.co.opentone.fsol.beeprovider");

try {
    Cursor cur = getContentResolver().query(contentUri,
            null, null, new String[] { keyword }, null);
                :
```

呼び出し側では
条件のみ指定

title	content	date	place
会議 A	超重要な会議	2013-04-29	会議室 A
会議 B	定例会	2013-04-30	大会議室
会議 C	グダグダな会議	2013-05-01	給湯室

このコードの
脆弱性はなんだろう？？？



title	content	date	place
会議Ａ	超重要な会議	2013-04-29	会議室Ａ
会議Ｂ	定例会	2013-04-30	大会議室
会議Ｃ	グダグダな会議	2013-05-01	給湯室

```
String keyword = "a%' OR '%' = '";
Uri contentUri =  Uri.parse(
        "content://jp.co.opentone.fsol.beeprovider");

try {
    Cursor cur = getContentResolver().query(contentUri, null,
            null, new String[] { keyword }, null);
                :
```

title	content	date	place
会議 A	超重要な会議	2013-04-29	会議室 A
会議 B	定例会	2013-04-30	大会議室
会議 C	グダグダな会議	2013-05-01	給湯室

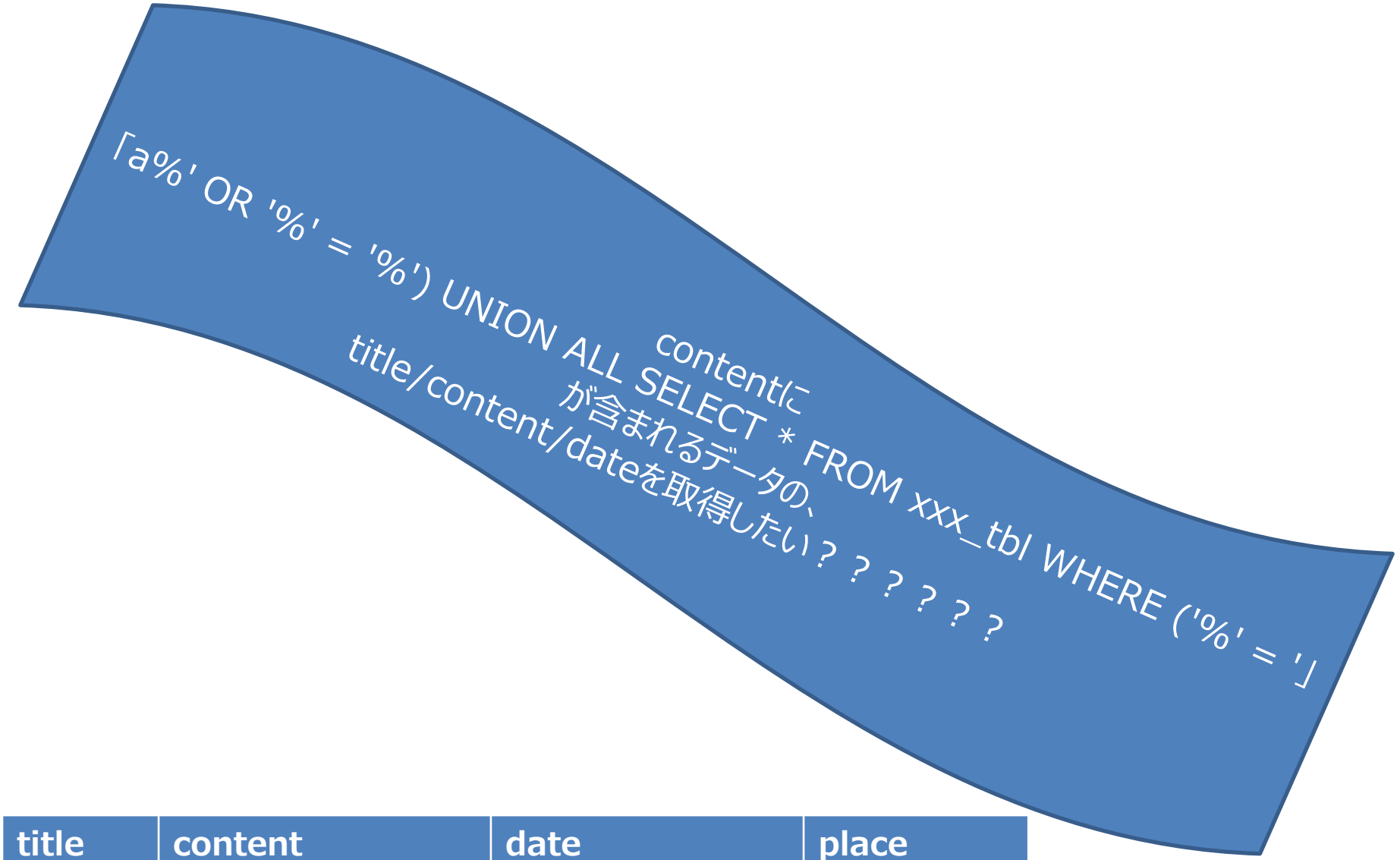
実際に発行されたQuery

```
SELECT title, content, date
  FROM schedule_tbl
  WHERE (content LIKE '%会議%')
```

```
SELECT title, content, date
  FROM schedule_tbl
  WHERE (content LIKE '%a%' OR '%' = '%')
```

SQLインジェクション

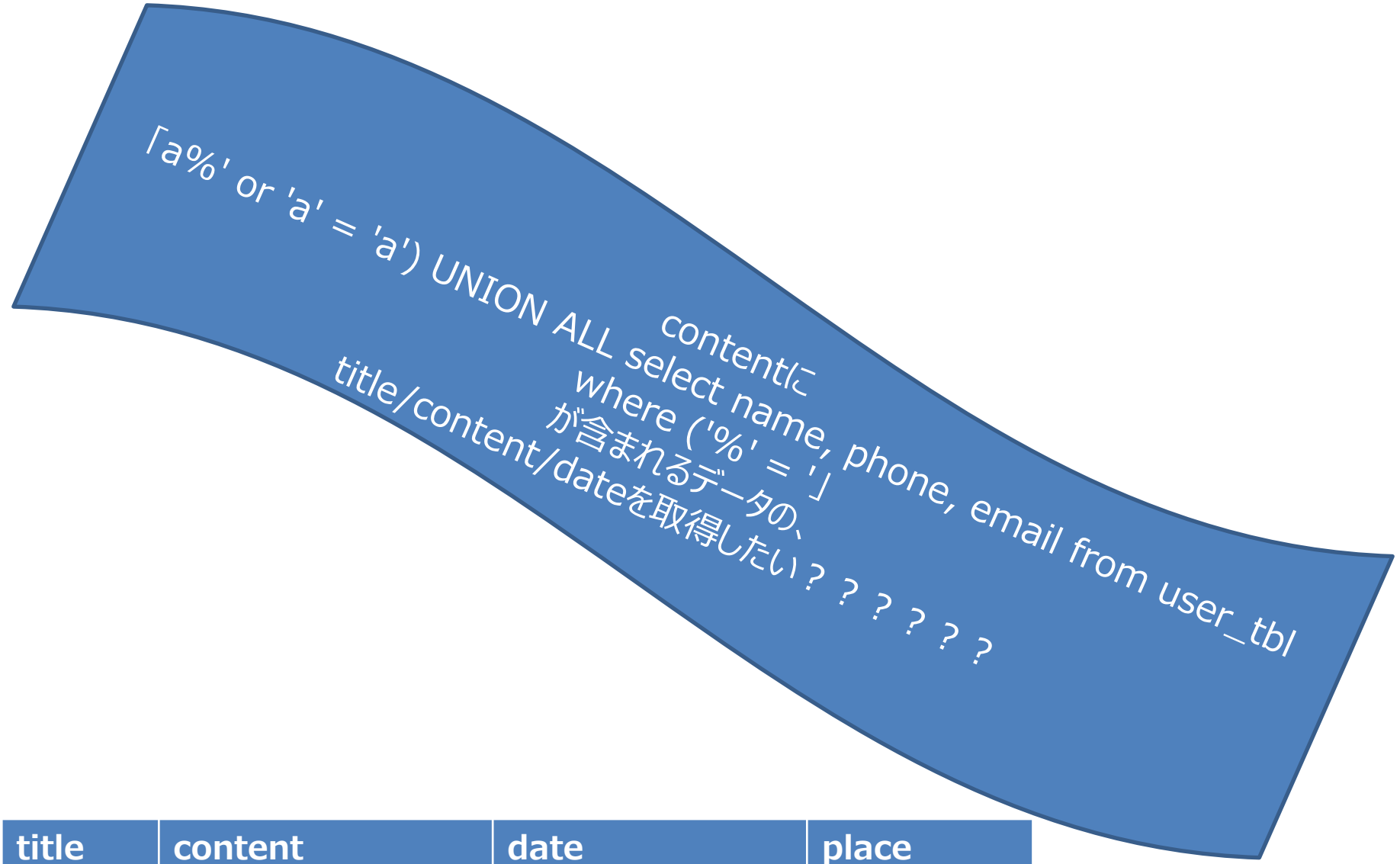
- データベースへの問い合わせや操作を行うプログラムに不正なキーワードを与える
  - データベースを改ざん
  - 不正に情報を入手



title	content	date	place
会議Ａ	超重要な会議	2013-04-29	会議室Ａ
会議Ｂ	定例会	2013-04-30	大会議室
会議Ｃ	グダグダな会議	2013-05-01	給湯室

```
String keyword = "a%' OR '%' = '%') UNION ALL SELECT * FROM
xxx_tbl WHERE ('%' = '";
Uri contentUri =  Uri.parse(
        "content://jp.co.opentone.fsol.beeprovider");

try {
    Cursor cur = getContentResolver().query(contentUri, null,
            null, new String[] { keyword }, null);
                :
```

title	content	date	place
会議Ａ	超重要な会議	2013-04-29	会議室Ａ
会議Ｂ	定例会	2013-04-30	大会議室
会議Ｃ	グダグダな会議	2013-05-01	給湯室



title	content	date	place
会議A	超重要な会議	2013-04-29	会議室A
会議B	定例会	2013-04-30	大会議室
会議C	グダグダな会議	2013-05-01	給湯室

```
String keyword = "a%' or 'a' = 'a') UNION ALL select name, phone,
email from user_tbl where ('%' = '";
Uri contentUri =  Uri.parse(
        "content://jp.co.opentone.fsol.beeprovider");

try {
    Cursor cur = getContentResolver().query(contentUri, null,
            null, new String[] { keyword }, null);
                :
```

title	content	date	place
会議 A	超重要な会議	2013-04-29	会議室 A
会議 B	定例会	2013-04-30	大会議室
会議 C	グダグダな会議	2013-05-01	給湯室

```
String keyword = "a%' or 'a' = 'a') UNION ALL select name, phone,
email from user_tbl where ('%' = '";
Uri contentUri =  Uri.parse(
        "content://jp.co.opentone.fsol.beeprovider");

try {
    Cursor cur = getContentResolver().query(contentUri, null,
            null, new String[] { keyword }, null);
                :
```

name	phone	email
武藤敬司	090-0000-xxxx	keiji.muto@three.musketeers.com
蝶野正洋	090-0001-xxxx	masahiro.chono@three.musketeers.com
橋本真也	090-0002-xxxx	shinya.hashimoto@three.musketeers.com

対応

- プレースホルダーの利用
- エラーメッセージの制御
- 条件キーワードのエスケープ

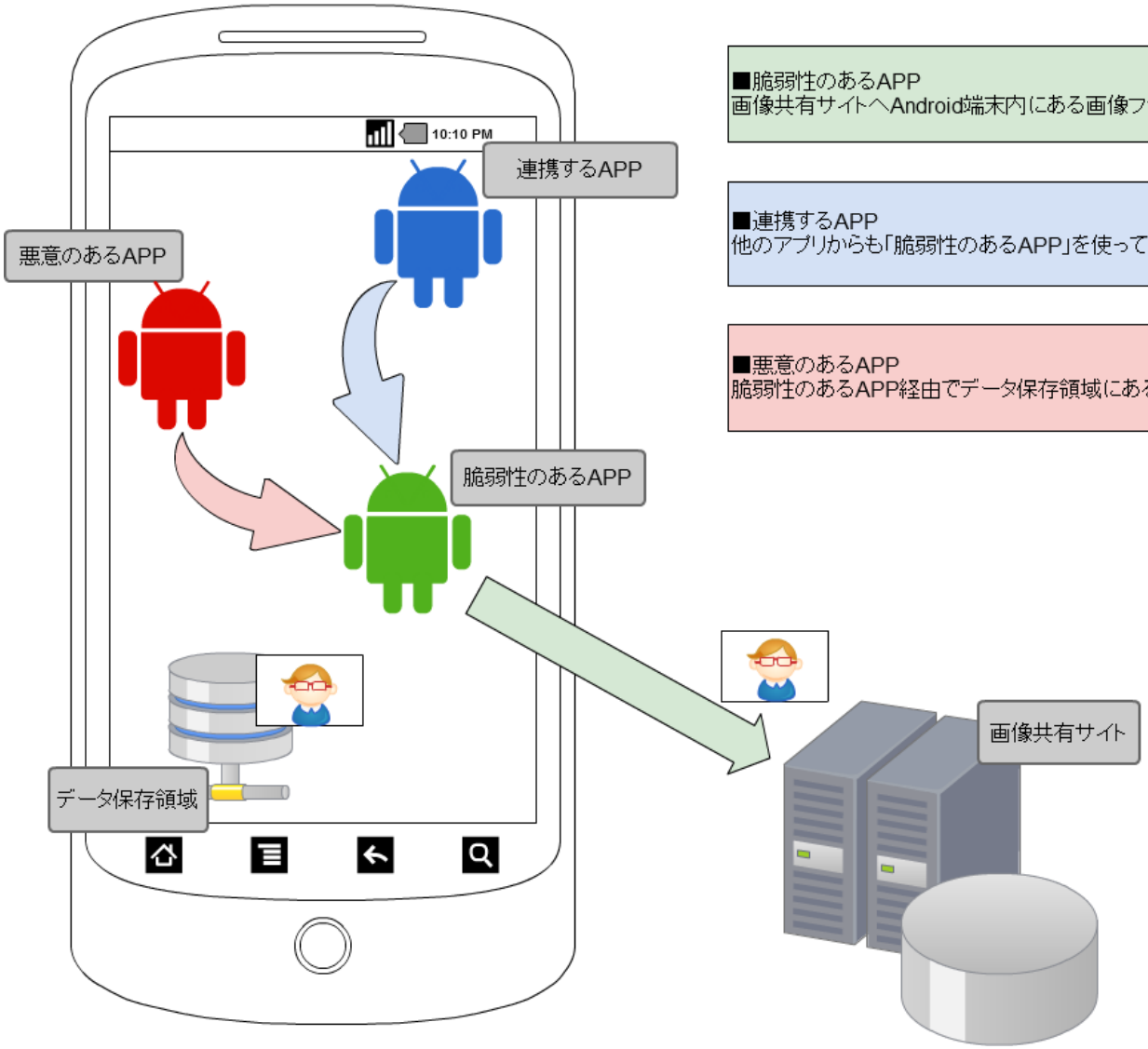
```
try {
    SQLiteDatabase db = databaseHelper.getReadableDatabase();
    SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
    qb.setTables("schedule_tbl");
    projection = new String[] { "title", "content", "date" };
    String condition = selectionArgs[0].replaceAll("'", "''");
    selection = "content LIKE '%" + condition + "%'";
    selectionArgs = null;
    Cursor c = qb.query(db, projection, selection, selectionArgs,
            null, null, null);

    return c;
} catch (Exception e) {
    Log.e(e.getClass().getName(), e.getMessage());
    return null;
}
```

キーワードのエスケープ

エラーメッセージの制御

title	content	date	place
会議A	超重要な会議	2013-04-29	会議室A
会議B	定例会	2013-04-30	大会議室
会議C	グダグダな会議	2013-05-01	給湯室

事例アプリ（其ノ二）



■脆弱性のあるAPP
画像共有サイトへAndroid端末内にある画像ファイルをアップロードする

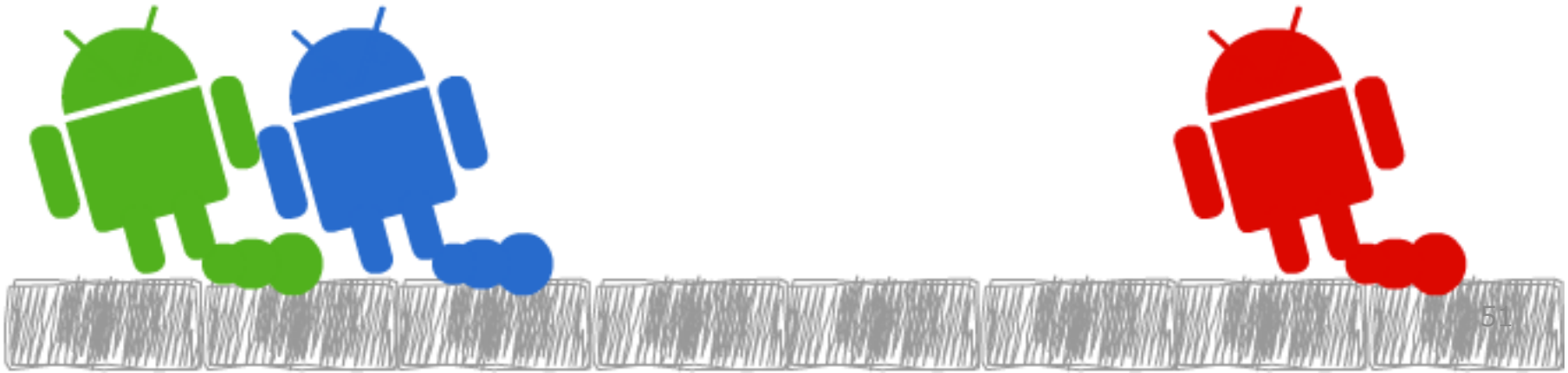
■連携するAPP
他のアプリからも「脆弱性のあるAPP」を使って画像共有サイトへ画像ファイルをアップロードすることができる

■悪意のあるAPP
脆弱性のあるAPP経由でデータ保存領域にある画像ファイルを外部サイトにアップロードする

脆弱性のある箇所

アクセスを許可したAPPだけに呼び出されたい…
予期せぬAPPから呼び出されてしまう…

「Intentに関するアクセス制限の不備が原因！！」



Intent

- Android特有のアプリケーション間通信手段
  - 独立したアプリケーション間での連携が可能となる
- 明示的Intentと暗黙的Intent
  - 特性を正しく理解しないと、セキュリティインシデントを引き起こしてしまう

発生する被害

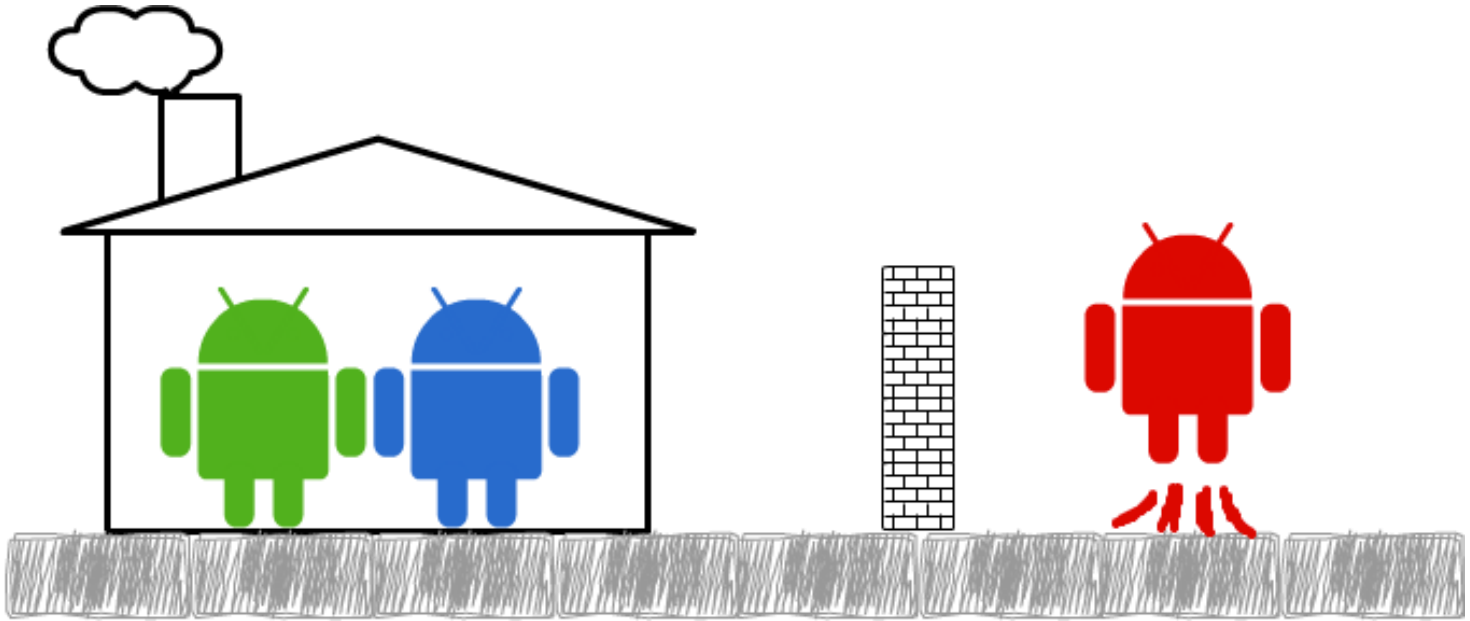
悪意のあるAPPが、公開したくないプライベートな写真までアップロードしてしまう…

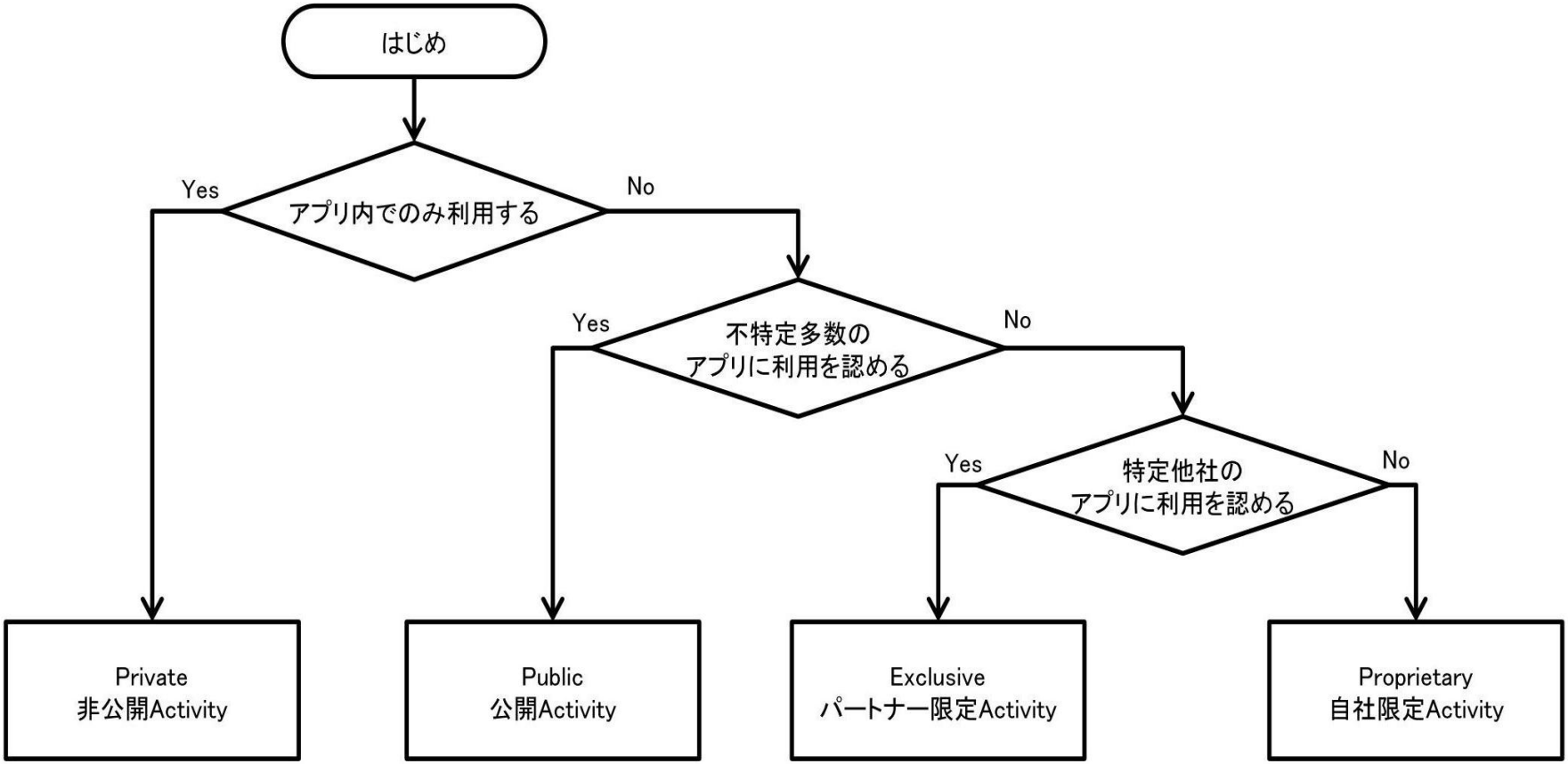
「画像ファイルの流出！！」

対策

Intentのアクセス制御を許可したいAPPのみにする。

「activityがどのように利用されるかにより、
activityが抱えるリスクや適切な防御手段が異なる」





出典：
Androidアプリのセキュア設計・セキュアコーディングガイド
http://www.jssec.org/dl/android_securecoding.pdf

インテントを使用するときの注意点

- 指定したアクションどおりに処理されない可能性
- インテントのエクストラ情報が解析される可能性
- アクティビティマネージャのログからインテントが解析される可能性
- アンドロイドの標準アプリケーションと連携するときの注意点

最後に

マルウェアを防ぐには

- 信頼のおける所からインストール
- パーミッションを理解する

アプリ作成時の脆弱性を回避するには

- ContentProviderやIntentの設定を正しく行う
- 通常のAPPと同様な脆弱性も留意する

かたじけない