KOF 2012

Androidセキュアコーディングのツボ

2012年11月10日

熊谷 裕志 hiroshi.kumagai@jpcert.or.jp

JPCERT CC®

アジェンダ

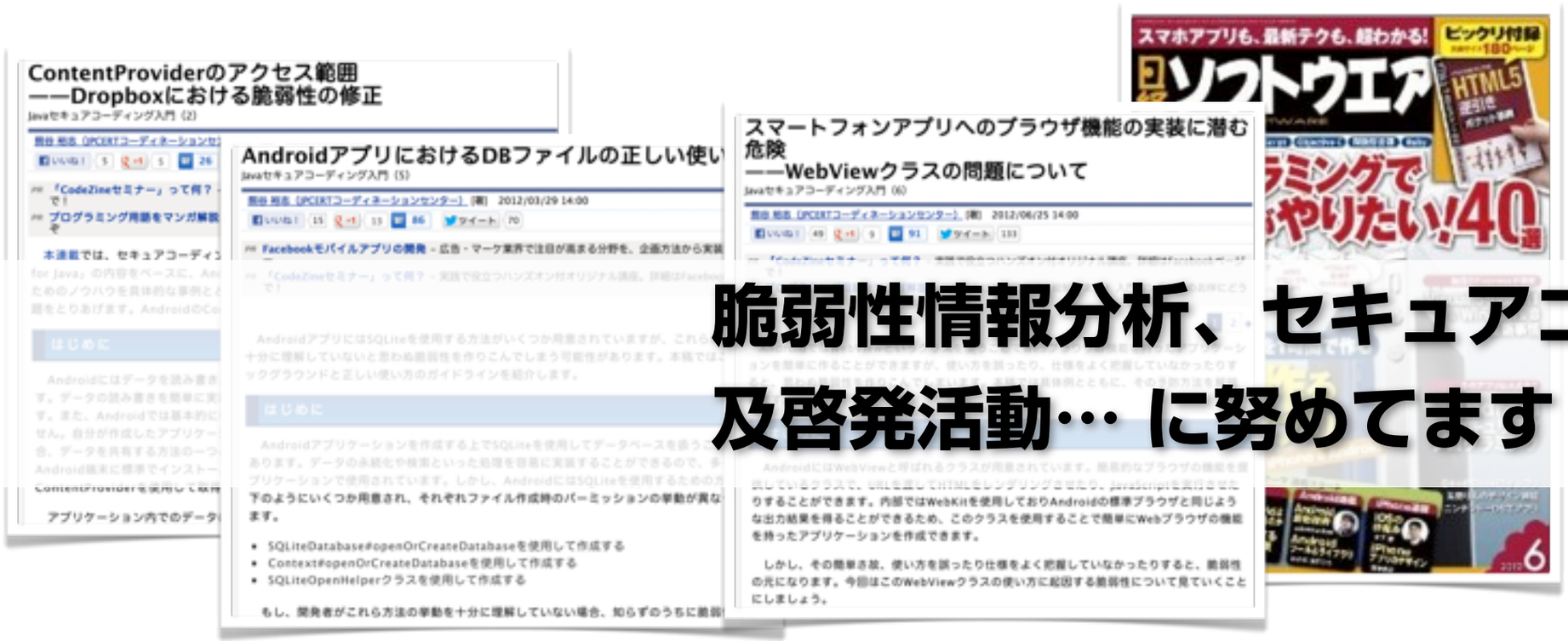
0. JPCERT/CC とは

1. DBファイル作成時の注意

2. WebViewの脆弱性

自己紹介

JPCERT/CC
情報流通対策グループ 解析チーム
リードアナリスト　熊谷裕志



脆弱性情報分析、セキュアコーディング普及啓発活動… に努めてます

0. JPCERT/CC とは

1. DBファイル作成時の注意

2. WebViewの脆弱性

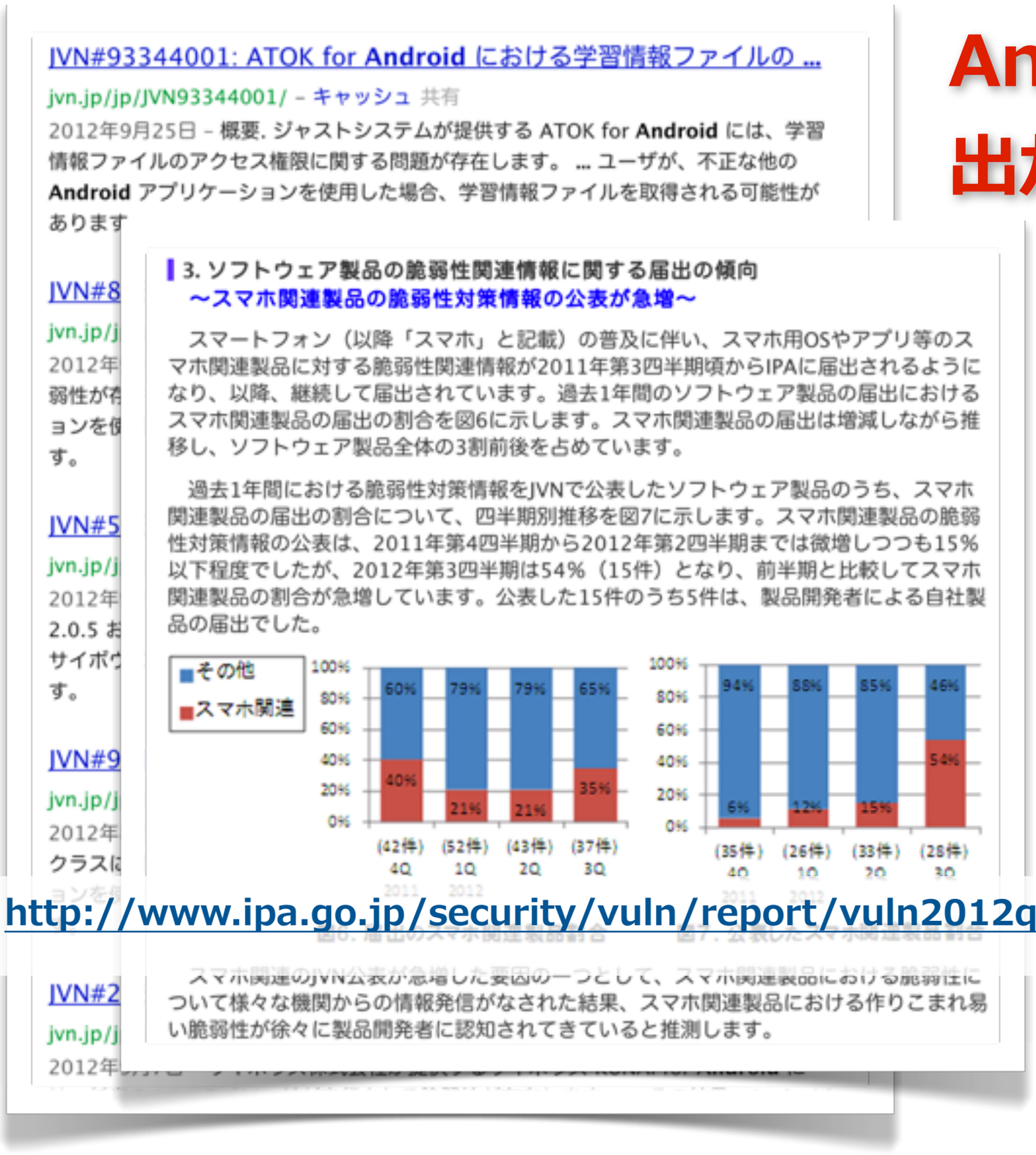
JPCERT CC®
Japan Computer Emergency Response Team Coordination Center
JPCERTコーディネーションセンター

JPCERT Coordination Center

日本における情報セキュリティ対策活動の向上に取り組んでいる組織

JPCERT/CCの主な活動

Androidアプリの脆弱性



http://www.ipa.go.jp/security/vuln/report/vuln2012q3.html#t03

Androidアプリの脆弱性の届出が増えてきている

- アクセス制限不備
- WebView関連
- ログ出力関連
- etc...

0. JPCERT/CC とは

1. DBファイル作成時の注意

2. WebViewの脆弱性

記事を書きました

CodeZine: Androidアプリにおける DBファイルの正しい使い方

http://codezine.jp/article/detail/6495

AndroidアプリにおけるDBファイルの正しい使い方

Javaセキュアコーディング入門（5）

熊谷 裕志（JPCERTコーディネーションセンター） [著] 2012/03/29 14:00

いいね！ 15 +1 13 86 ツイート 70

PR プログラミング用語をマンガ解説 - 初学者向けに専門用語を絵解きます！ 入門コンテンツのお伴にどうぞ

PR Facebookモバイルアプリの開発 - 広告・マーケ業界で注目が高まる分野を、企画方法から実装までカバー

1 2

AndroidアプリにはSQLiteを使用する方法がいくつか用意されていますが、これらの挙動を十分に理解していないと思わぬ脆弱性を作りこんでしまう可能性があります。本稿ではこのバックグラウンドと正しい使い方のガイドラインを紹介します。

はじめに

Androidアプリケーションを作成する上でSQLiteを使用してデータベースを扱うことは多々あります。データの永続化や検索といった処理を容易に実装することができるので、多くのアプリケーションで使用されています。しかし、AndroidにはSQLiteを使用するための方法が以下のようにいくつか用意され、それぞれファイル作成時のパーミッションの挙動が異なっています。

- SQLiteDatabase#openOrCreateDatabaseを使用して作成する
- Context#openOrCreateDatabaseを使用して作成する
- SQLiteOpenHelperクラスを使用して作成する

もし、開発者がこれら方法の挙動を十分に理解していない場合、知らずのうちに脆弱性を作り込んでしまいます。

DBファイル作成時におけるファイルパーミッションの問題について

DBファイルを作成する方法

- SQLiteDatabase#openOrCreateDatabase
- Context#openOrCreateDatabase
- SQLiteOpenHelperクラス

問題となる例

SampleContentProvider.java

～中略～

```
SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(new File(
    "/data/data/" + getContext().getPackageName() + "/databases/",
    DATABASE), null);

final String sql = "" +
    "CREATE TABLE IF NOT EXISTS `items`(" +
    " `_id` INTEGER PRIMARY KEY AUTOINCREMENT," +
    " `title` VARCHAR(255), `description` TEXT," +
    " `level` INTEGER, `identifier` TEXT, `link` TEXT," +
    " `datetime` VARCHAR(255), `created_at` INTEGER" +
    ")";

db.execSQL(sql);
db.close();
```

SQLiteDatabase#openOrCreateDatabase
を使用している

DBファイルのパーミッションが不適切

SQLiteDatabase#openOrCreateDatabaseを使用してDBファイルを作成すると…

```
-rw-r--r-- app_60   app_60       10240 2012-10-01 15:50 rss.db
-rw-r--r-- app_60   app_60           0 2012-10-01 15:50 rss.db-journal
-rw-rw---- app_60   app_60       32768 2012-10-01 17:16 webview.db-shm
-rw-rw---- app_60   app_60           0 2012-10-01 17:16 webview.db-wal
-rw-rw---- app_60   app_60         168 2012-09-12 18:41 webviewCookiesChromium.db
-rw-rw---- app_60   app_60           0 2012-09-12 18:46 webviewCookiesChromiumPrivate.db
```

作成されたDBファイルのパーミッションが不適切になってしまう

なぜなのか

- パーミッションを設定する引数がない
- 生成されるDBファイルのパーミッションが644になる
- SQLiteDatabase#dbopenを呼び出している
  - ネイティブのメソッドを呼び出している
    - android_database_SQLiteDatabase.cpp
      - dbopenメソッド
        - sqlite3_open_v2関数でファイルを作成している
        - パーミッションを660にすることはできない

Context#openOrCreateDatabase

- パーミッションを設定する引数がある

- 内部では
  - SQLiteDatabase#openOrCreateDatabaseを呼び出す
  - 呼び出した後にsetFilePermissionsFromModeを使用して、指定されたパーミッションに変更している

- 設定可能パーミッション
  - Context.MODE_PRIVATE
  - Context.MODE_WORLD_READABLE
  - Context.MODE_WORLD_WRITEABLE

SQLiteOpenHelperクラス

- 内部では
  - Context#openOrCreateDatabaseを呼び出している

- パーミッションは設定不可
  - Context.MODE_PRIVATE固定

SQLiteOpenHelperクラスの内部

```
boolean success = false;
SQLiteDatabase db = null;
if (mDatabase != null) mDatabase.lock();
try {
    mIsInitializing = true;
    if (mName == null) {
        db = SQLiteDatabase.create(null);
    } else {
        db = mContext.openOrCreateDatabase(mName, 0, mFactory, mErrorHandler);
    }
```

Context#openOrCreateDatabaseを呼び出している

まとめ

- ~~SQLiteDatabase#openOrCreateDatabase~~
- Context#openOrCreateDatabase
- SQLiteOpenHelperクラス

0. JPCERT/CC とは

1. DBファイル作成時の注意

2. WebViewの脆弱性

WebViewの脆弱性とは?

Full Disclosure: Android Multiple Vulnerabilities

http://seclists.org/fulldisclosure/2012/Feb/111

2012年2月に

Full Disclosureで公表された

記事を書きました

CodeZine: スマートフォンアプリへのブラウザ機能の実装に潜む危険 -- WebViewクラスの問題について

http://codezine.jp/article/detail/6618

スマートフォンアプリへのブラウザ機能の実装に潜む危険
——WebViewクラスの問題について

Javaセキュアコーディング入門 (6)

熊谷 裕志（JPCERTコーディネーションセンター） [著] 2012/06/25 14:00

いいね！ 49 +1 9 91 ツイート 133

PR 「CodeZineセミナー」って何？ - 実践で役立つハンズオン付オリジナル講座。詳細はFacebookページで！

PR Facebookモバイルアプリの開発 - 広告・マーケ業界で注目が高まる分野を、企画方法から実装までカバー

1 2

AndroidではWebViewというクラスを使うことでWebブラウザの機能を持ったアプリケーションを簡単に作ることができますが、使い方を誤ったり、仕様をよく把握していなかったりすると、思わぬ脆弱性を作りこんでしまいます。本稿では具体例とともに、その予防方法を解説します。

はじめに

AndroidにはWebViewと呼ばれるクラスが用意されています。簡易的なブラウザの機能を提供しているクラスで、URLを渡してHTMLをレンダリングさせたり、JavaScriptを実行させたりすることができます。内部ではWebKitを使用しておりAndroidの標準ブラウザと同じような出力結果を得ることができるため、このクラスを使用することで簡単にWebブラウザの機能を持ったアプリケーションを作成できます。

しかし、その簡単さ故、使い方を誤ったり仕様をよく把握していなかったりすると、脆弱性の元になります。今回はこのWebViewクラスの使い方に起因する脆弱性について見ていくことにしましょう。

WebViewクラスとJavaScript

WebViewクラスを使用した場合、注意しなければならないのはJavaScriptを有効にした場合です。デフォルトではJavaScriptの機能は無効にされていますが、

- Full Disclosureが元ネタ
- プラスα

- fileスキーム
- addJavascriptInterface

問題となるアプリの実装は？

fileスキームを含むURLを受け取りそれをWebViewクラスで表示している

問題となる例：脆弱なアプリ

WebViewActivity.java

```
final String url = getIntent().getStringExtra("url");
 ～中略～
wSettings.setJavaScriptEnabled(true);
webView.loadUrl(url);
```

端末内にあるHTMLファイルのURLをアプリに投げます

file:///mnt/sdcard/exploit.html

問題となる例：攻撃者が用意するHTML

file:///mnt/sdcard/exploit.html

```
<script>
  var xmlhttp = new XMLHttpRequest();

  xmlhttp.open(
      'GET',
      'file:///data/data/xxxxxxxx/databases/webview.db',
      false);
  xmlhttp.send(null);
  var ret = xmlhttp.responseText;

  document.getElementById('result').innerHTML = ret;
</script>
```

このHTMLを表示するアプリのファイルパス

ファイルの中身を表示

解決方法：fileスキーム

例：ページを表示しない

```
final String url = getIntent().getStringExtra("url");

String loadUrl = "about:blank";
if (!url.startsWith("file:")) {
    loadUrl = url;
}
```

fileスキームが含まれているかチェックする

例： Javascriptを無効にする

```
final String url = getIntent().getStringExtra("url");

wSettings.setJavaScriptEnabled(false);
if (!url.startsWith("file:")) {
    wSettings.setJavaScriptEnabled(true);
}
```

fileスキームが含まれているかチェックする

API Level 16(Android 4.1)から

Fileスキームの読み込みに関するメソッドが追加

WebSettings#setAllowFileAccessFromFileURLs
WebSettings#setAllowUniversalAccessFromFileURLs

public abstract void **setAllowFileAccessFromFileURLs** (boolean flag)

Sets whether JavaScript running in the context of a
to access content from other file scheme URLs. To e
therefore secure policy, this setting should be disabl
is ignored if the value of `getAllowUniversalAcce`

The default value is true for API level `ICE_CREAM_SA`
for API level `JELLY_BEAN` and above.

Parameters

flag whether JavaScript running in the context
allowed to access content from other file s

public abstract void **setAllowUniversalAccessFromFileURLs** (boolean flag)

Since: API Level 16

Sets whether JavaScript running in the context of a file scheme URL should be allowed to access content from any origin. This includes access to content from other file scheme URLs. See `setAllowFileAccessFromFileURLs(boolean)`. To enable the most restrictive, and therefore secure policy, this setting should be disabled.

The default value is true for API level `ICE_CREAM_SANDWICH_MR1` and below, and false for API level `JELLY_BEAN` and above.

Parameters

flag whether JavaScript running in the context of a file scheme URL should be allowed to access content from any origin

http://developer.android.com/intl/ja/reference/android/webkit/WebSettings.html#setAllowFileAccessFromFileURLs(boolean)

まとめ：fileスキーム

URLにfileスキームが含まれている場合は、Javascriptを無効にするか、ページを表示しないように制限すること

問題となるアプリの実装は？

WebView#addJavascriptInterfaceを使用している

問題となる例：addJavascriptInterface

WebViewActivity.java

```
final String url = getIntent().getStringExtra("url");
  ～中略～
wSettings.setJavaScriptEnabled(true);
webView.addJavascriptInterface(new JsObject(this), "jsObject");
webView.loadUrl(url);
```

JsObject.java

```
private Context mContext;

public JsObject(Context context) {
    // TODO Auto-generated constructor stub
    mContext = context;
}
```

問題となる例：addJavascriptInterface

Javascriptを使用してContextを取得されてしまう

```
var myclass = jsObject.getClass();
var field = myclass.getDeclaredField('mContext');
field.setAccessible(true);
var context = field.get(jsObject);
var p = context.getPackageManager();
var l = p.getInstalledApplications(8192);
var s = l.toString()
document.write(p.toString());
document.write("<br />");
document.write(s.substr(0, 100));
document.write("<br />");
for (var x = 0; x < l.size(); x++) {
    document.write("<br />");
    document.write(l.get(x));
}
document.write("<br />");
```

端末にインストールされている
アプリの一覧を取得する例

問題となる例：Context

Contextを持っていなければ
いいんじゃない？

問題となる例：クラスローダーを取得できる

Javascriptからクラスローダーを取得することもできる

```
var classLoader = myclass.getClass().getClassLoader();
var c = classLoader.loadClass(
            'android.telephony.TelephonyManager');
document.write(c.toString());
document.write("<br />");
```

このサンプルではTelephonyManagerをロードしているが、このクラスではなく別の、**とあるクラスをロードするとContextが取得できてしまう**

@goroh_kunの発表資料 http://ierae.co.jp/uploads/webview.pdf

解決方法：addJavascriptInterface

現状、解決方法はありません

まとめ：addJavascriptInterface

addJavascriptInterfaceは

使用するな

参考1

Android Security Overview

Androidプラットフォームにおけるセキュリティの概要

http://source.android.com/tech/security/index.html

Designing for Security

セキュリティを念頭に置いたアプリ開発の指針をまとめた文書

http://developer.android.com/guide/practices/security.html

Security and Permissions

パーミッションの仕組みを正しく使う方法について解説

http://developer.android.com/guide/topics/security/permissions.html

参考2

- JSSEC『Androidアプリのセキュアコーディングガイド』

  http://www.jssec.org/dl/android_securecoding.pdf

- Java セキュアコーディングスタンダード CERT/Oracle版

  https://www.jpcert.or.jp/java-rules/

- 『Android Security 安全なアプリケーションを作成するために』タオソフトウェア株式会社