JPCERT CC®

JEB Plugin 開発チュートリアル 第1回

－JEB Pluginとは－
構造、UIからの情報取得と設定方法
を修得する

一般社団法人JPCERTコーディネーションセンター

目次

- 第0回 JEBとは？
- **第1回 JEB Pluginとは**
  - **1. JEB Pluginの使い方**
  - **2. JEB Pluginの構造**
  - **3. JEBのUIを利用するためのAPI**
  - **4. ViewとSignature**
- 第2回 DEXファイルの構造を理解する
  - 1. DEXファイルの構造
  - 2. jeb.api.dex
  - 3. クロスリファレンス
- 第3回 バイトコードについての理解
  - 1. CodeItem
- 第4回 JEB PluginからASTを扱う

1. JEB PLUGINの使い方

JEB Pluginとは

- 用意されているAPIを使って、JEBの機能を拡張し、解析作業を効率化するためのスクリプト
  - JavaまたはPythonで記述する
  - JEBはJavaで実装されており、PythonのスクリプトはJythonというJava Runtime上でPythonを実行するフレームワークを使用
    - Nativeコードを含むようなPythonライブラリは使用できない
    - Pure Pythonのライブラリは?
- JEBに同梱されているPlugin
  - JavaとPythonのサンプルPlugin
  - ライブラリを識別するPlugin
  - Signatureを生成するPlugin
- 次のURLからも既成Pluginをダウンロードできる
  - http://www.android-decompiler.com/download.php
- API Reference
  - http://www.android-decompiler.com/apidoc/

JEB Pluginの中で使用できるAPI

- jeb.api
- jeb.api.ui
- jeb.api.dex
- jeb.api.ast

これらAPIを使用して、JEBを自動操作したり、DEXファイルの操作やASTを使用してアプリを解析することが可能。

JEB Pluginの指定方法

- Plugin実行までの流れ
  1. JEBを起動する
  2. 解析対象となるAndroidアプリをJEBで開く
  3. Pluginを指定して実行する

- 実行するPluginを指定するには3つの方法がある
  - —[File] → [Run Script…]
    - 実行したいファイルを選択し実行する
  - —[File] → [Run last Script] (Ctrl(⌘) + i)
    - 最後に実行したPluginを実行する
  - —[Action] → [Custom Actions]から選択
    - JEBディレクトリ/plugins 以下にあるファイルが[Custom Actions](次のページ参照)に一覧表示される

Custom Actions

- JEBの[Action]メニューの中の [Custom Actions]に一覧表示されるPlugin
  - ショートカットキーを割り当てることが可能
  - メニューに一覧表示されるので、Pluginの実行が簡単
- Custom Actionsに表示されるPluginの置く場所は環境設定で変更可能
- マジックコメント
  - Pluginの先頭行に、”?”で始まるコメントを
    - Custom Actionsの表示やショートカットなどをカスタマイズ
  - key=value, …のような記法で記述する
- 記述内容
  - name, author, shortcut, help

```
#? name=Signature Generator, shortcut=Ctrl+Shift+S, author=Nicolas Falliere, help=Create binary signatures for library code recognition
```

マジックコメントは、JEB起動時に読み込まれるので、途中で変更しても反映されない
JEBを再起動する必要がある
※ Plugin自体は毎回読み込まれるので挙動は変更出来る

2. JEB PLUGINの構造

- jeb.api.IScript
- jeb.api.JebInstance

JEB Pluginの記述方法

- PythonまたはJavaで記述する
  - —Pythonを使用した例

```python
from jeb.api import IScript

class SamplePluginPython(IScript):

  def run(self, jeb):
      jeb.print("This line is generated by a Python plugin")
```

  - —Javaを使用した例

```java
import jeb.api.IScript;
import jeb.api.JebInstance;

public class SamplePluginJava implements IScript {

    public void run(JebInstance jeb) {
        jeb.print("This line is generated by a Java plugin");
    }
}
```

JEB Plugin例

- まずは次のPluginを書いてみよう

```
from jeb.api import IScript

class Hello(IScript):

  def run(self, jeb):
      jeb.print('Hello World!!')
```

—Hello World!!を標準出力に出力するするサンプルPlugin

- 注意
  - —クラス名とファイル名は同一にする
    - class Hello → Hello.py

jeb.api.IScript クラス

```
from jeb.api import IScript

class Hello(IScript):

  def run(self, jeb):
      jeb.print('Hello World!!')
```

- JEBのPluginの基底クラス
- 全てのJEB Pluginはこのクラスを継承して作成
- Pluginを実行すると、IScriptクラスを継承したクラスのrunメソッドがJEBから呼び出される
  - —IScript.run()をオーバーライドし、Pluginの処理を記述する
- runメソッドにはJebInstanceオブジェクトが引数で渡される

jeb.api.JebInstanceとは

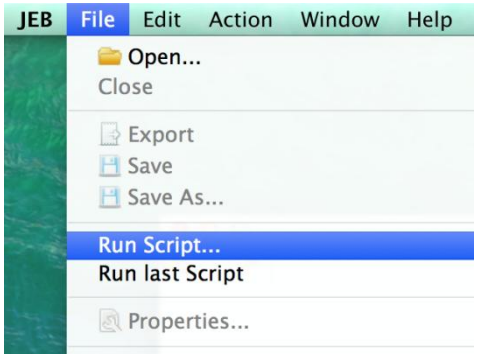
- JEBのアプリケーションインスタンス
- IScript.runメソッドの引数としてPluginに渡される
- JebInstance主な機能
  - JEBの操作
    - load, save, close, exit
  - 情報の取得/設定
    - ユーザ情報、APIバージョン
    - コメント
    - デコンパイルソースコード
    - View
    - Dex、 AST
    - rename

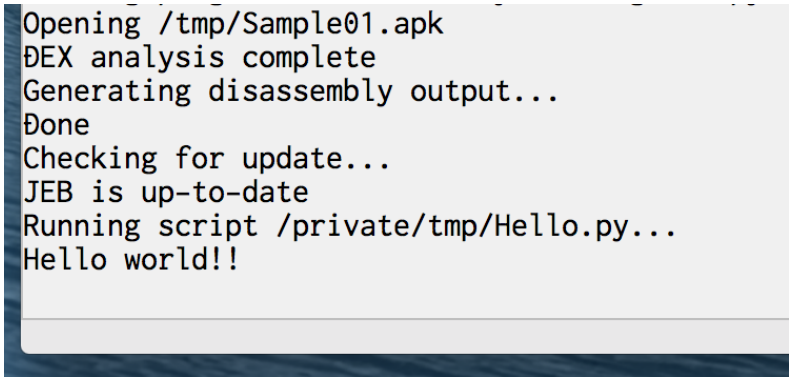
例題1

- JEB Pluginを登録して実行する
  - —Hello.py

[例題1] JEB Pluginを登録して実行する

- 課題
  - 先の「JEB Plugin例」で書いたPluginをJEBに登録して実行する
- Pluginはどこにおいても良い
  - [File] → [Run Script…]でファイル選択する
- JEBの[File]から[Run Script…]で実行する





- 標準出力はJEBのコンソールウィンドウに出力される
  - JEBには出力用のメソッドとしてJebInstance.printメソッドが用意されているが、代わりにPythonのprint文を使用することもできる

例題2

- すべてのコメントをコンソールに出力するPluginを作成する
  - —jeb.api.Commentクラスの使い方を理解する

[例題2] すべてのコメントをコンソールに出力するPluginを作成する

- 課題
  - —すべてのコメントをコンソールに出力する
- 期待する出力結果
  - jeb.api.Commentから取得できる情報を全て出力する
- 注意
  - —事前に、JEBを使用してapkファイルに複数のコメントを挿入してからPluginを実行する
- ヒント
  - —JebInstance.getAllComments()
    - jeb.api.Commentの配列が返される
    - どんなクラスなのか、APIリファレンスで確認しよう

例題2の解答例

- AllComments.py

例題2で作ったPluginの実行

- 解答例にならってPluginを自作し、アプリSample01.apkに対して実行してみてください

```
Running script /Users/vul-an/bin/jeb/plugins/AllComments.py...
sig:Lcom/example/sample01/MainActivity;->onCreate(Landroid/os/Bundle;)V, offset:6
          comment: "R.layout.activity_main"
sig:Lcom/example/sample01/MainActivity;->onCreateOptionsMenu(Landroid/view/Menu;)Z,
offset:8
          comment: "R.menu.main"
```

例題3

- Custom Actionsに設定してみよう
  —マジックコメントの書き方を理解する

[例題3] Custom Actions に設定してみよう

- 例題2で作成したPluginを Custom Actions に設定する
  - —P.6の解説を参考に
  - —使い方
    - Pluginをpluginsディレクトリに置く
    - Pluginにマジックコメントを書く
      - —名前とショートカットを指定
    - JEBを再起動
    - 設定したショートカットで実行する

[解答例] Custom Actionsに設定してみよう

```
#? name=All Comments, shortcut=Shift+Ctrl+C, author=vulan,
help=Show all comments,
```

- Pluginの先頭に"?"で始まるコメントを記述することで、ショートカットキーを割り当てることができる
- この例では、ShiftキーとCtrlキーとCを押すことで、JEBのGUIからこのPluginを実行することができるようになる

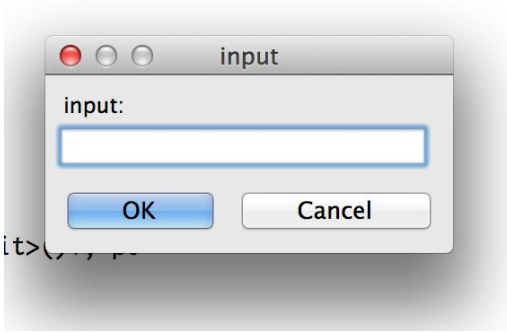
3. JEBのUIを利用するためのAPI

jeb.api.uiクラス

- JebUI
  - JEBのUIをコントロールしているクラス
    - メッセージボックスの表示
      - JebUI.displayQuestionBox, displayMessageBox
    - Viewの取得
      - JebUI.getView(View.Type)
    - Viewの表示設定
      - JebUI.forcusView(View.Type), isVisibleView(View.Type)
    - ステータスバーの取得・表示
    - デコンパイルの実行
- View
  - JEBに用意されているAssemblyやJava、Manifestなどの各Viewを定義している
- CodePosition
  - CodeViewの位置を表すクラス
- Enums (定数値)
  - BottunGroup
    - OK、OK_CANCEL、YES_NO、YES_NO_CANCEL
  - IconType
    - ERROR、INFORMATION、QUESTION、WARNING
  - View.Type
    - ASSEMBLY、CLASS_HIERARCHY、CONSOLE、JAVA、MANIFEST、NOTES

JEBのメッセージボックス

- JebUI.displayQuestionBox
  —ユーザからの入力を求めるメッセージボックスを表示する



- JebUI.displayMessageBox
  —メッセージボックスを表示する
  —ButtonGroupTypeでボタンの種類を選択できる

例題4

- メッセージボックスの表示
  - —JebUI.displayQuestionBox()
    JebUI.displayMessageBox() を理解する

[例題4] メッセージボックスを表示する

以下の機能を実装してみよう

- displayQuestionBoxを使ってユーザ入力を求める
- 入力結果をdisplayMessageBoxで表示する
  - displayMessageBoxは、IconTypeとButtonGroupTypeを指定する必要がある

例題4の解答例

- MessageBox.py

[解説] メッセージボックスを表示する

```
from jeb.api import IScript

class MessageBox(IScript):
    def run(self, jeb):
        ui = jeb.getUI()

        ret = ui.displayQuestionBox("Who", "Who are you ?", "")
        print(ret)

        ret = ui.displayMessageBox("return", "retun value:" + ret,
                ui.IconType.ERROR, ui.ButtonGroupType.YES_NO_CANCEL)
        print(ret)
```

- ui.displayQuestionBoxの返り値にはユーザが入力したデータが入っている
- ui.displayMessageBoxの返り値にはユーザがクリックしたボタンの番号(int)が入っている
  - 0(cancel), 1(ok), 2(yes), 3(no)

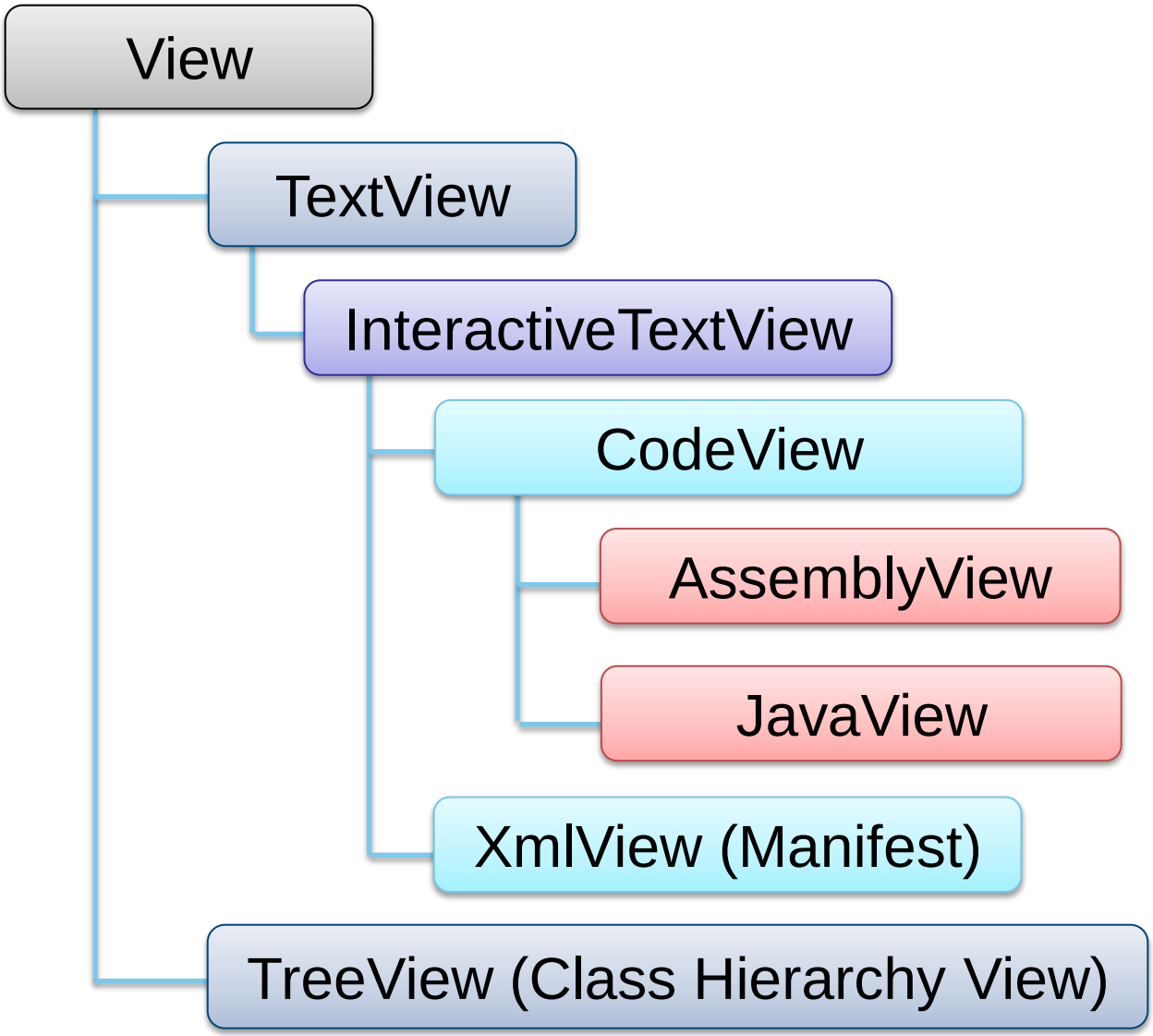
4. VIEWとSIGNATURE

ViewとSignature

- JEBには各用途に合ったViewが用意されている
  - —ディスアセンブリしたコードを表示するAssembly View
  - —デコンパイルされたJavaのコードを表示するJava View
  - —AndroidManifest.xmlを表示するManifest View
  - —クラス階層構造を表示するClass Hierarchy View

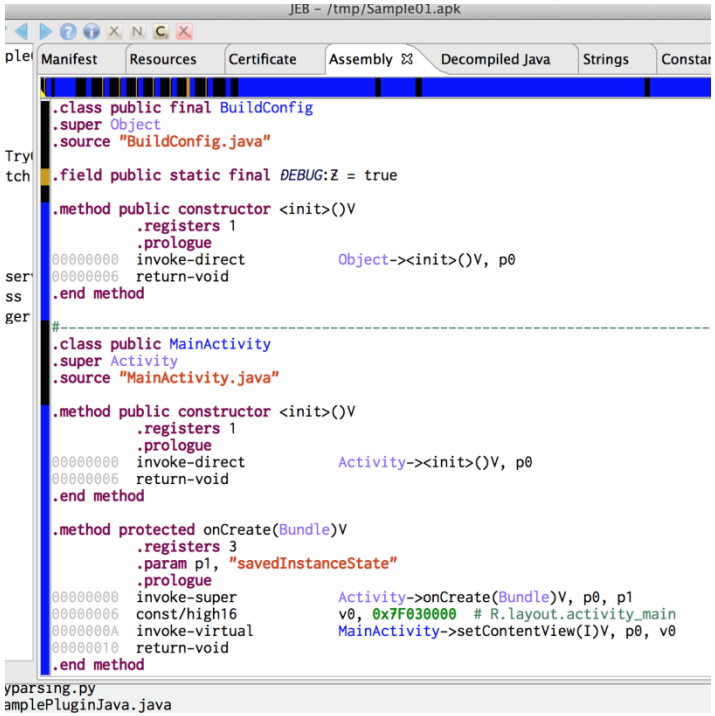
- JEBにはDEX内にあるItem(class, method, field)を一意に特定するための文字列(Signature)がある
  - —Signatureを使用して、デコンパイル

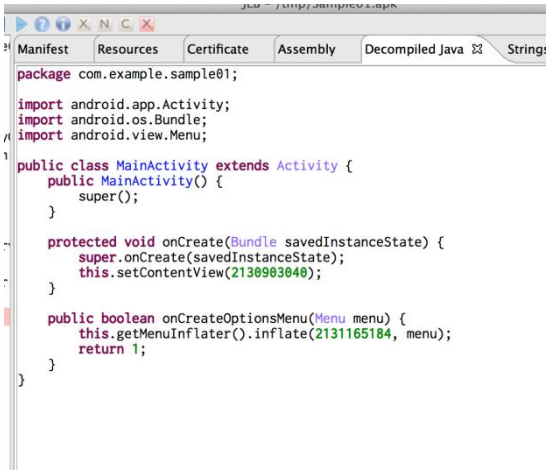
Viewクラスの継承関係



View

- Assembly View
  - —DEXファイルをディスアセンブリしたコードが表示されるビュー
- Java View
  - —デコンパイルされたJavaのコードが表示されるビュー





この他にもManifest ViewやNotes Viewなどがあり、JEB Pluginで操作できるViewは、View.Typeで定義されている。

Viewの操作

- 共通
  - refresh
- CodeView(Assembly, Java)
  - getCodePosition
    - キャレット位置の取得
  - CodePosition.getSignature()
- AssemblyView
  - setCodePosition
    - キャレット位置の指定
- InteractiveView
  - getActiveItem
    - キャレットが当たっているItemの名前

Signature (partial_sig)

- CodePosition.getSignatureで得られる文字列
- DEXのItem(class, method, field)を一意に特定する
  - —これをつかうことで、いろんな情報が取得できる
- クラス
  - —**Lcom/example/SomeObject;**
    - “L”+パッケージパス+クラス名 + ”;”
- メソッド
  - —**Lcom/example/SomeObject;->getSome()String**
    - クラス名 + ”->” +メソッド名+ (引数) + 型
- フィールド
  - —**Lcom/example/SomeObject;->FieldName:String**
    - クラス名 + ”->” + フィールド名 “:” + 型

Signature(partial_sig) 用途

- JebInstanceクラス
  - —decompileClass(String), decompileMethod(String)
    - 引数にはSignatureを渡す
    - デコンパイルした文字列が返される
  - —getClass, getMethod
    - ASTの取得
- JebUI.decompileClass()
  - —指定したメソッドをデコンパイルして、Java Viewにフォーカス
- AssemblyView.setCodePosition()
  - —CodePositionのコンストラクタの引数にもつかえる
  - —それでsetCodePositionが出来る

例題5

- Decompile Codeを表示する
  - —Viewの切り替えを理解する
  - —Signatureの使い方を理解する
  - —Pluginからデコンパイルする方法を理解する

[例題5] Decompile Codeを表示する

- 課題
  - —Assembly Viewでキャレット位置のメソッドを
    デコンパイルして、コンソールに出力してみよう
- 期待する出力結果

```
[Lcom/example/contentprovider/MainActivity;->asyncTask()V]
private void asyncTask() {
    new com.example.contentprovider.MainActivity$RssTask(this, ((android.app.Activity)this)).execute(new java.lang.Object[0]);
}
```

- ヒント
  - —例えばこのような流れで出力できる
    1. JebUIの取得
    2. Viewの取得
    3. CodePositionの取得
    4. Signatureの取得
    5. デコンパイル結果の取得　（from JebInstance)
    6. コンソールへの表示

例題5の解答例

- DecompileMethod.py

[解説] Decompile Codeを表示する

```
# get assembly view
view = ui.getView(View.Type.ASSEMBLY)
```

1.

```
# get signature from caret position
msig = view.getCodePosition().getSignature()
```

2.

```
print("[%s]" % msig)
print(jeb.decompileMethod(msig))
```

3.

1. まずui.getView()メソッドでAssemblyViewを取得する
2. getCodePosition()メソッドで現在のキャレット位置のCodePositionを取得し、getSignature()でそこのSignatureを取得する
3. decompileMethod()メソッドに取得したSignatureを渡すことで、デコンパイルされたコードが取得できる

例題6

- ManifestView -> AssemblyViewを便利にする
  - Viewの切り替えとAndroidManifest.xmlのパースを理解する

[例題6] ManifestView→AssemblyViewを便利にする

- 課題
  - —Manifest Viewでキャレット位置にあるActivityのname属性を取得して、そのクラスのAssembly Viewにジャンプする Pluginを作成してみよう
- ヒント
  - —Signature
    - “.”で始まるものは、パッケージ名を足す必要がある
    - それ以外はそのまま使える
    - SignatureはLxxx/yyy/Zzz;の形式
  - —キャレット位置のアイテムの取得
    - View.getActiveItem()
  - —クラス一覧
    - JebInstance.getDex().getClassSignatures()で取得できる
  - —パッケージ名
    - XML Parser (xml.etree.ElementTree)

例題6の解答例

- ManifestJump.py

[解説] ManifestView -> AssemblyViewを便利にする

```
from xml.etree.ElementTree import *
```

- ManifestViewに表示されているXML(AndroidManifest.xml)をパースする必要があるのでXML ParserをImportする

```
# get package name
xml = jeb.getManifest()
root = fromstring(xml)
package = root.attrib['package']
```

1.

```
# determine target names
if target.startswith('.'):
    target = package + target
elif target.find(u'.') == -1:
    target = package + '.' + target
```

2.

```
target = "L%s;" % target.replace('.', '/')
```

3.

1. getManifest()メソッドを使用してAndroidManifest.xmlを取得し、XML Parserを使用してパッケージ名を取得する
2. クラス名が"."で始まっている場合は"."の前にパッケージ名を追加する
3. "."を"/"に変換し先頭に"L"を追加して、Signatureを作る

[解説] ManifestView -> AssemblyViewを便利にする

```
# if target class is included in dex, jump to target class
if target in jeb.getDex().getClassSignatures(False):
    print('jump to ' + target)
    ui.focusView(View.Type.ASSEMBLY)
    asm_view = ui.getView(View.Type.ASSEMBLY)
    asm_view.setCodePosition(CodePosition(target))
else:
    print('jump target is not found')
```

- 作成したSignatureがクラスのSignatureだった場合、focusView()メソッドでAssemblyViewに切り替えて、該当するコードにキャレット位置を移動させる