

# Android App 竄改及防範措施

Jason Wang 王羿廷

艾斯冰殼股份有限公司

# About Me

王羿廷 Jason Wang

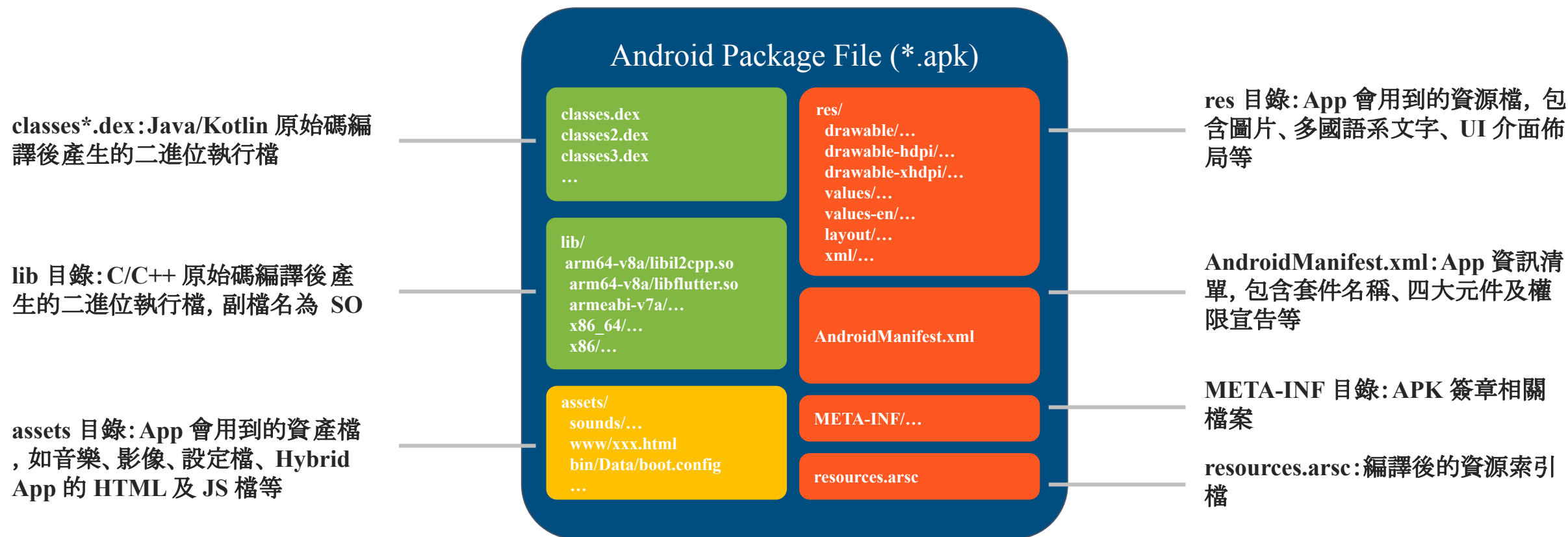
- ▶ 艾斯冰殼 ICEshell 創辦人暨執行長
- ▶ 專長
  - App 逆向工程
  - App 安全防護技術研發
- ▶ 研究成果
  - 台灣第一個國產 Android 加殼產品作者
  - 程式碼保護專利 M553450 發明人
  - 曾任 iThome CYBERSEC 講師
  - 曾任 CSA Taiwan Summit 講師



# Table of Contents

- APK 檔案結構分析
- DEX 檔反編譯
- SO 檔反編譯
- APK 修改及二次打包
- 防範措施

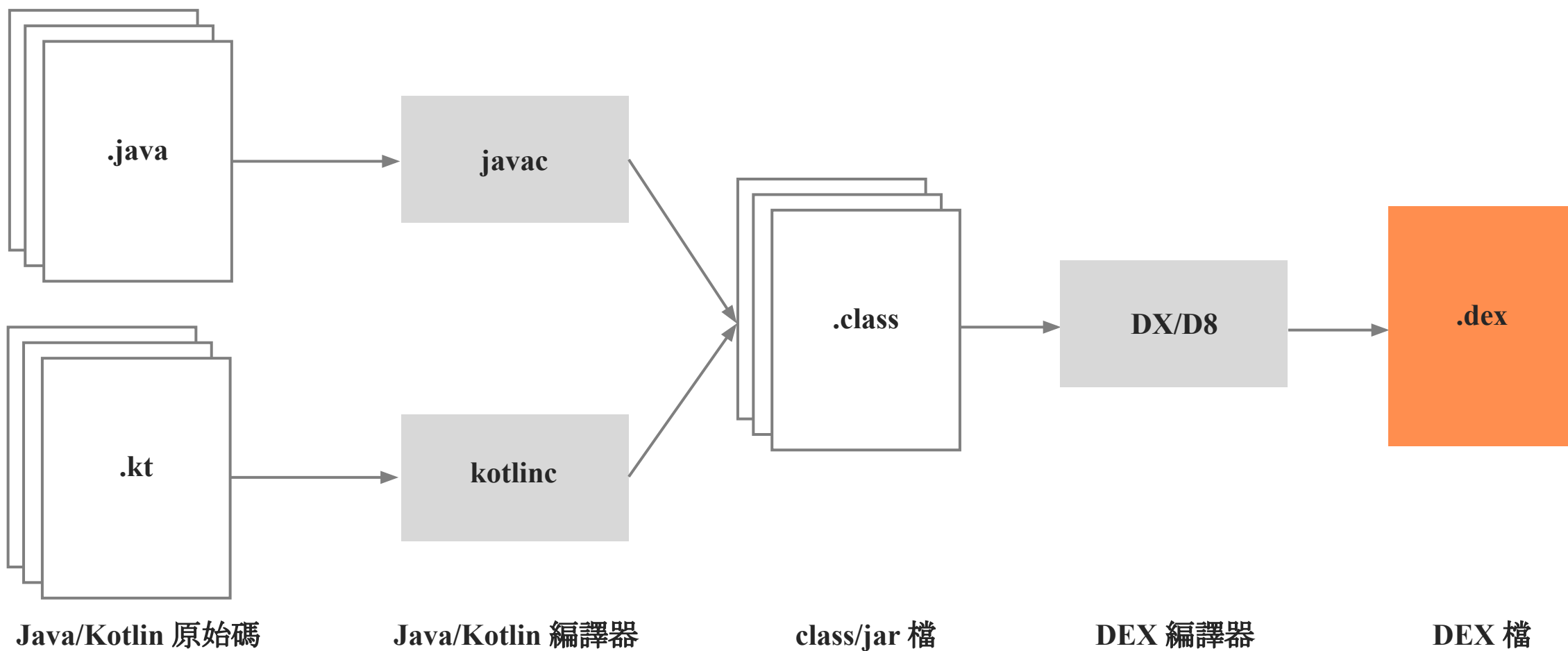
# APK 檔案結構





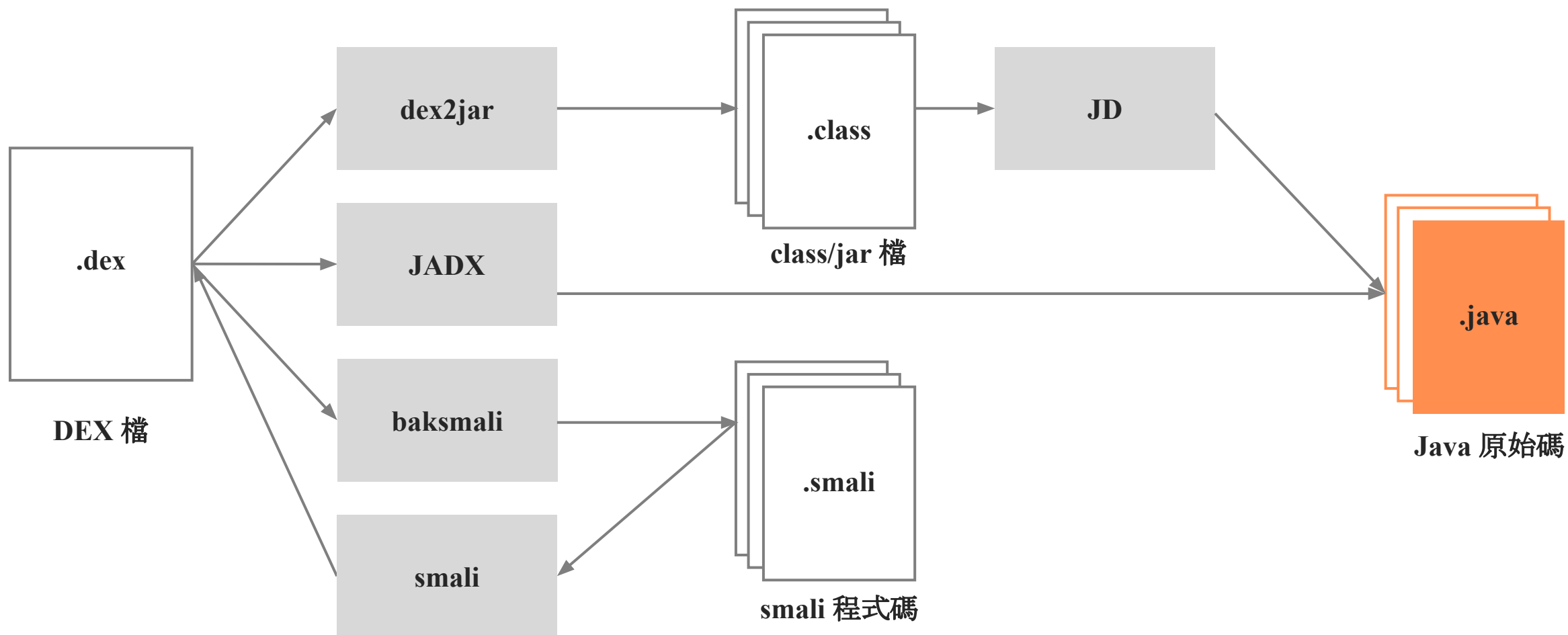
# DEX 檔反編譯

# DEX (Dalvik Executable) 檔的產生





# DEX 檔反編譯與重新編譯



# 逆向分析小技巧

如何從茫茫原始碼海中找出關鍵程式碼

---

1. 尋找 App 入口點
2. 搜尋字串
3. 搜尋系統 API 呼叫





逆向分析小技巧

# 尋找 App 入口點

---

在 AndroidManifest.xml 裡檢查 `<application>` 與 `<activity>` 元素：

1. 檢查 `<application>` 是否存在「`android:name`」屬性，若存在表示此 App 有自訂的 Application 類別，App 入口點即為該類別的 `attachBaseContext()` 或 `onCreate()` 方法。
2. 尋找包含 `category` 為「`android.intent.category.LAUNCHER`」的 `activity` 類別，此類別即為 App 的主要 `activity`，對於無自訂 Application 類別的 App，入口點即為主要 `activity` 類別的 `attachBaseContext()` 或 `onCreate()` 方法。

## 逆向分析小技巧

# 尋找 App 入口點

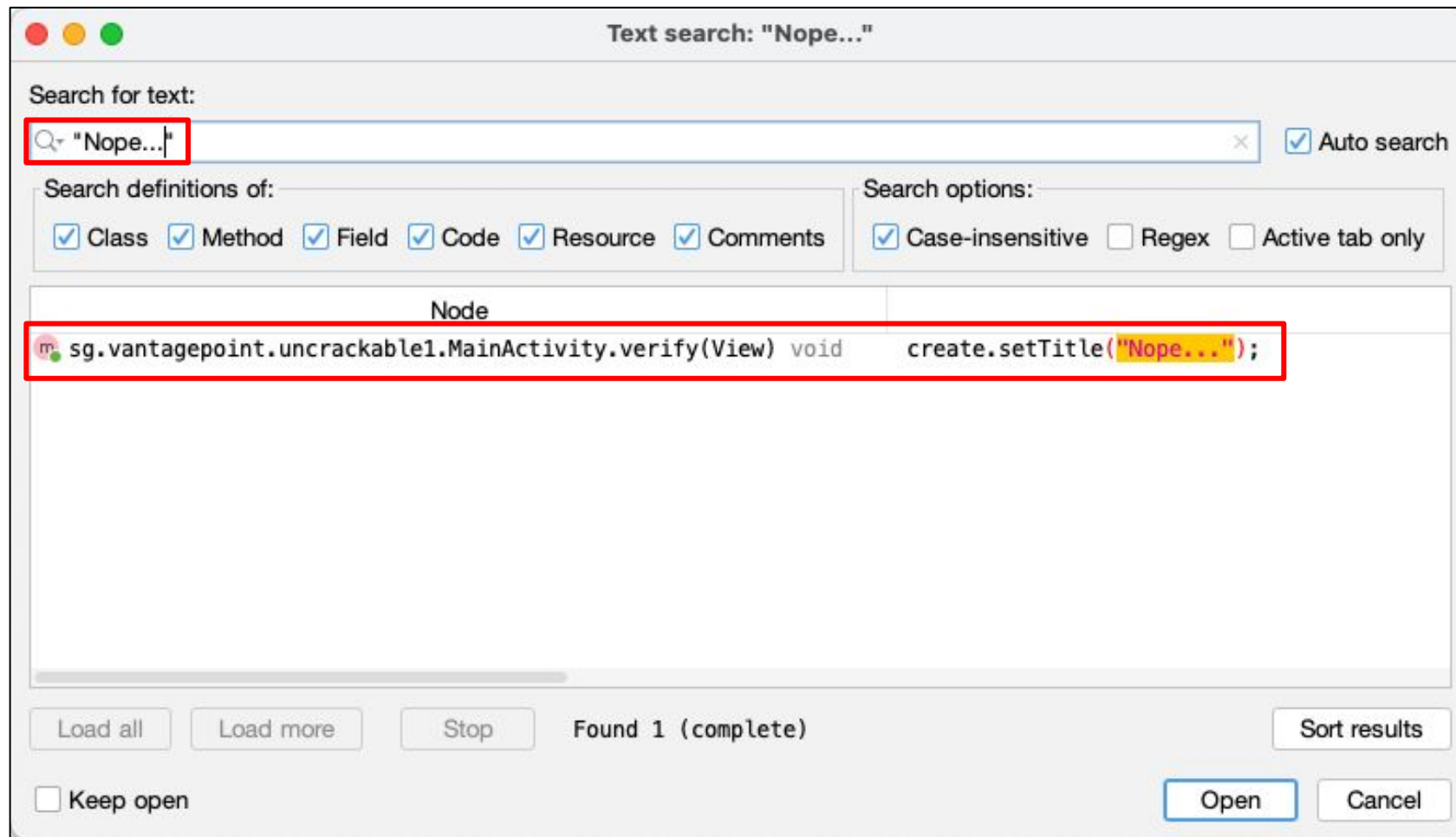
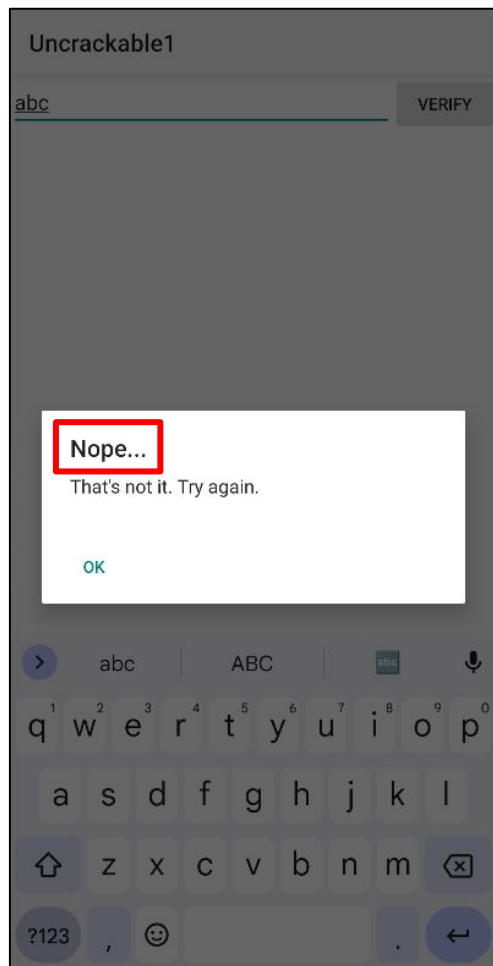
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="owasp.mstg.uncrackable1">
  <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="28"/>
  <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:allowBackup="true">
    <activity android:label="@string/app_name" android:name="sg.vantagepoint.uncrackable1.MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

1. `<application>` 裡無 `android:name` 屬性, 此 App 無自訂的 Application 類別, 故 App 入口點不在此處
2. `category` 為「`android.intent.category.LAUNCHER`」的 `<activity>` 為主要 activity, 查看其 `android:name` 屬性
3. 此 App 入口點即為「`sg.vantagepoint.uncrackable1.MainActivity.onCreate()`」

## 逆向分析小技巧

# 搜尋字串

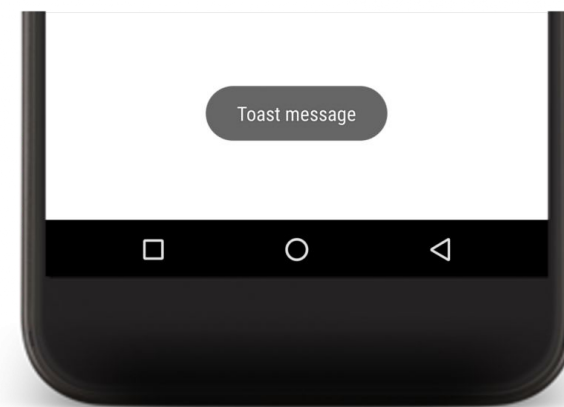
可在反編譯工具裡搜尋 App 出現過的錯誤訊息字串，快速定位到關鍵點



# 搜尋系統 API 呼叫

常見的系統 API 呼叫：

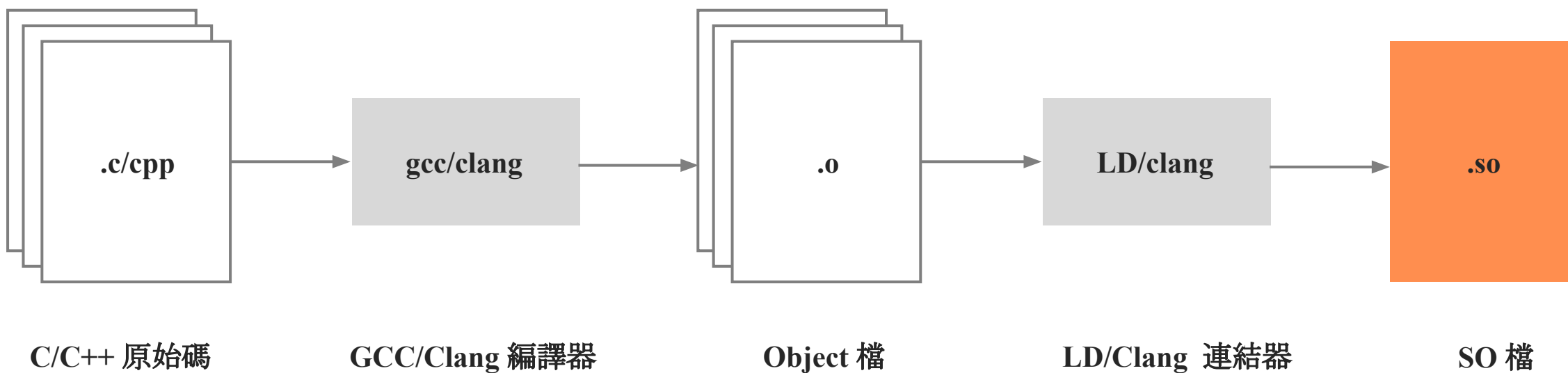
- 浮動式訊息：
  - `Toast.makeText.show()`
- 對話方塊：
  - `new AlertDialog.Builder().create().show()`
- 按鈕點擊事件：
  - `OnClickListener()`
- 關閉 App：
  - `System.exit()`
  - `KillProcess()`



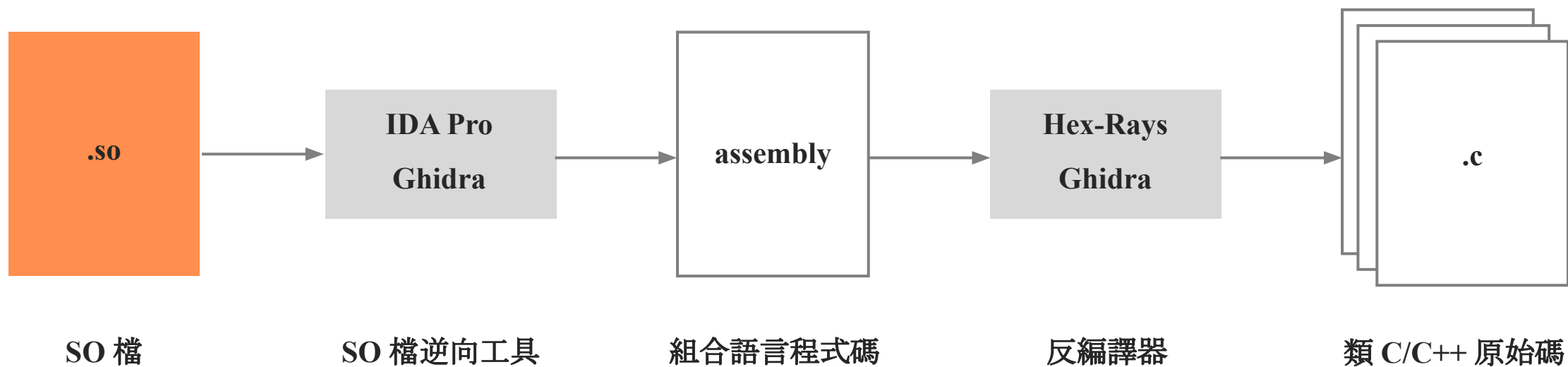


# SO 檔反編譯

# Android SO (Shared Object) 檔的產生



# SO 檔反編譯







# 逆向分析小技巧

如何從茫茫原始碼海中找出關鍵程式碼















---

1. 查看函數名稱
2. 搜尋字串
3. 搜尋系統 API 呼叫
4. 使用模擬執行工具分析程式行為

## 逆向分析小技巧

# 查看函數名稱







- 如果函數沒有特別宣告為 static, 編譯後的 SO 檔使用逆向工具可查看完整函數名稱
- 為了便於維護, 開發者通常會以該函數之功能為函數命名
- 看函數名稱即可了解該函數功用為何

 start	.text	000000000000E10
 nullsub_1	.text	000000000000E1C
 j_nullsub_1	.text	000000000000E20
 sub_E24	.text	000000000000E24
 rand(void)	.text	000000000000E4C
 main	.text	<b>000000000000F78</b>
 megalnit(void)	.text	000000000000FA4
 sub_10E0	.text	00000000000010E0
 <b>goodbye(void)</b>	.text	0000000000003080
 sub_31B0	.text	00000000000031B0
 sub_323C	.text	000000000000323C
 Java_sg_vantagepoint_uncrackable3_MainActivity_init	.text	0000000000003310
 Java_sg_vantagepoint_uncrackable3_MainActivity_baz	.text	0000000000003398
 Java_sg_vantagepoint_uncrackable3_CodeCheck_bar	.text	00000000000033A4

## 逆向分析小技巧

# 搜尋字串

- 可在反編譯工具裡搜尋 App 出現過的字串，快速定位到關鍵點
- 搜尋逆向工具名稱，可找出偵測逆向工具的程式碼

	.rodata:00000000000034C8	00000010	C	/proc/self/maps
	.rodata:00000000000034DA	00000006	C	frida
	.rodata:00000000000034E0	00000007	C	xposed
	.rodata:00000000000034E7	0000000D	C	UnCrackable3
	.rodata:00000000000034F4	00000023	C	Tampering detected! Terminating...
	.rodata:0000000000003517	0000002E	C	Error opening /proc/self/maps! Terminating...



















```
1 _int64 sub_30D0()
2 {
3     __int64 v0; // x23
4     const char *v1; // x2
5     __int64 v3; // [xsp+0h] [xbp-230h]
6
7     v0 = fopen("/proc/self/maps", "r");
8     if ( v0 )
9     {
10         do
11         {
12             while ( !fgets(&v3, 512LL, v0) )
13             {
14                 fclose(v0);
15                 usleep(500LL);
16                 v0 = fopen("/proc/self/maps", "r");
17                 if ( !v0 )
18                     goto LABEL_7;
19             }
20         }
21         while ( !strstr(&v3, "frida") && !strstr(&v3, "xposed") );
22         v1 = "Tampering detected! Terminating...";
23     }
24     else
25     {
26 LABEL_7:
27         v1 = "Error opening /proc/self/maps! Terminating...";
28     }
29     __android_log_print(2LL, "UnCrackable3", v1);
30     goodbye();
31     return sub_31B0();
32 }
```

## 逆向分析小技巧

# 搜尋系統 API 呼叫

常見的系統 API 呼叫：

- 字串相關: `strcmp()`、`strncmp()`、`strstr()`、`strlen()`、`strcpy()`、`strncpy()`...
- 檔案相關: `open()`、`fopen()`、`fgets()`、`fputs()`、`read()`、`write()`、`close()`、`fclose()`...
- 結束處理程序: `exit()`、`_exit()`、`kill()`、`tkill()`、`tgkill()`...

Address	Offset	Name
 0000000000015060		pthread_create
 0000000000015068		__cxa_finalize
 0000000000015070		ptrace
 0000000000015078		_exit
 0000000000015080		raise
 0000000000015088		fclose
 0000000000015090		__stack_chk_fail
 0000000000015098		fgets
 00000000000150A0		fork
 00000000000150A8		strncpy
 00000000000150B0		__android_log_print
 00000000000150B8		strstr
 00000000000150C0		usleep
 00000000000150C8		getppid
 00000000000150D0		malloc
 00000000000150D8		waitpid
 00000000000150E0		fopen
 00000000000150E8		pthread_exit

## 逆向分析小技巧

# 使用模擬執行工具分析程式行為

---

- 面對較難分析的 SO 檔，可使用模擬執行工具進行分析：
  - unidbg (<https://github.com/zhkl0228/unidbg>)
  - AndroidNativeEmu (<https://github.com/AeonLucid/AndroidNativeEmu>)
- 使用方式請參考 2022 台灣資安大會「Android NDK 程式 (.so檔) 逆向與防逆向」(<https://cyber.ithome.com.tw/2022/session-page/817>)



# APK 修改 及二次打包

# 修改版 APK 製作流程





## 修改範例

# OWASP MAS Crackmes: Android UnCrackable Level 1

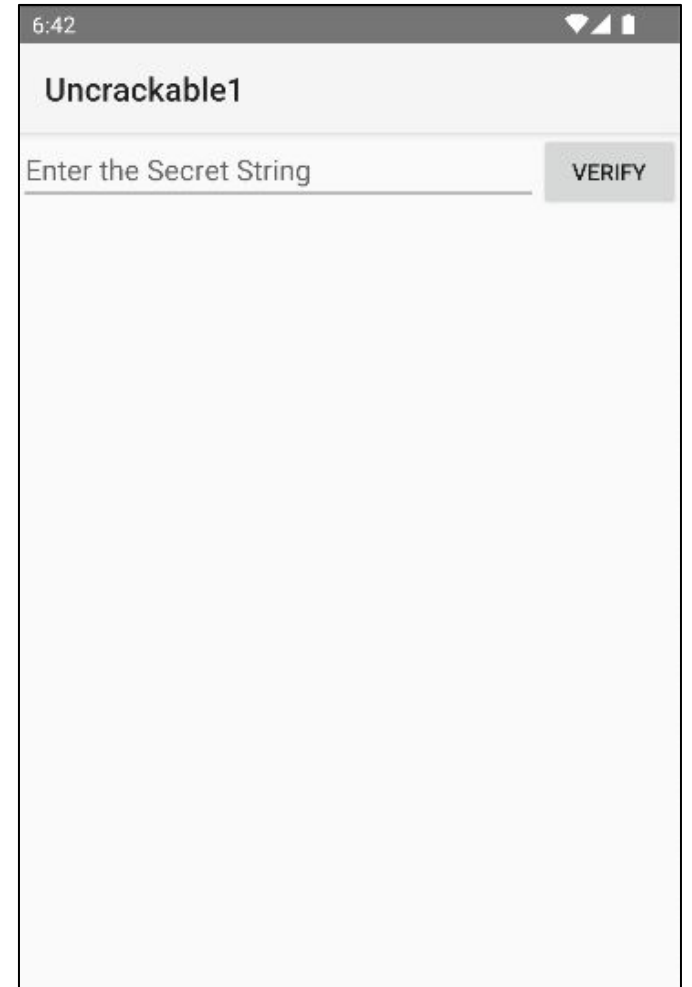
- 用於練習逆向工程的 App
- 由 OWASP Mobile Application Security 所提供
- 修改目標:輸入任何字元都能顯示驗證成功
- 下載網址:<https://mas.owasp.org/crackmes>

### UnCrackable Mobile Apps

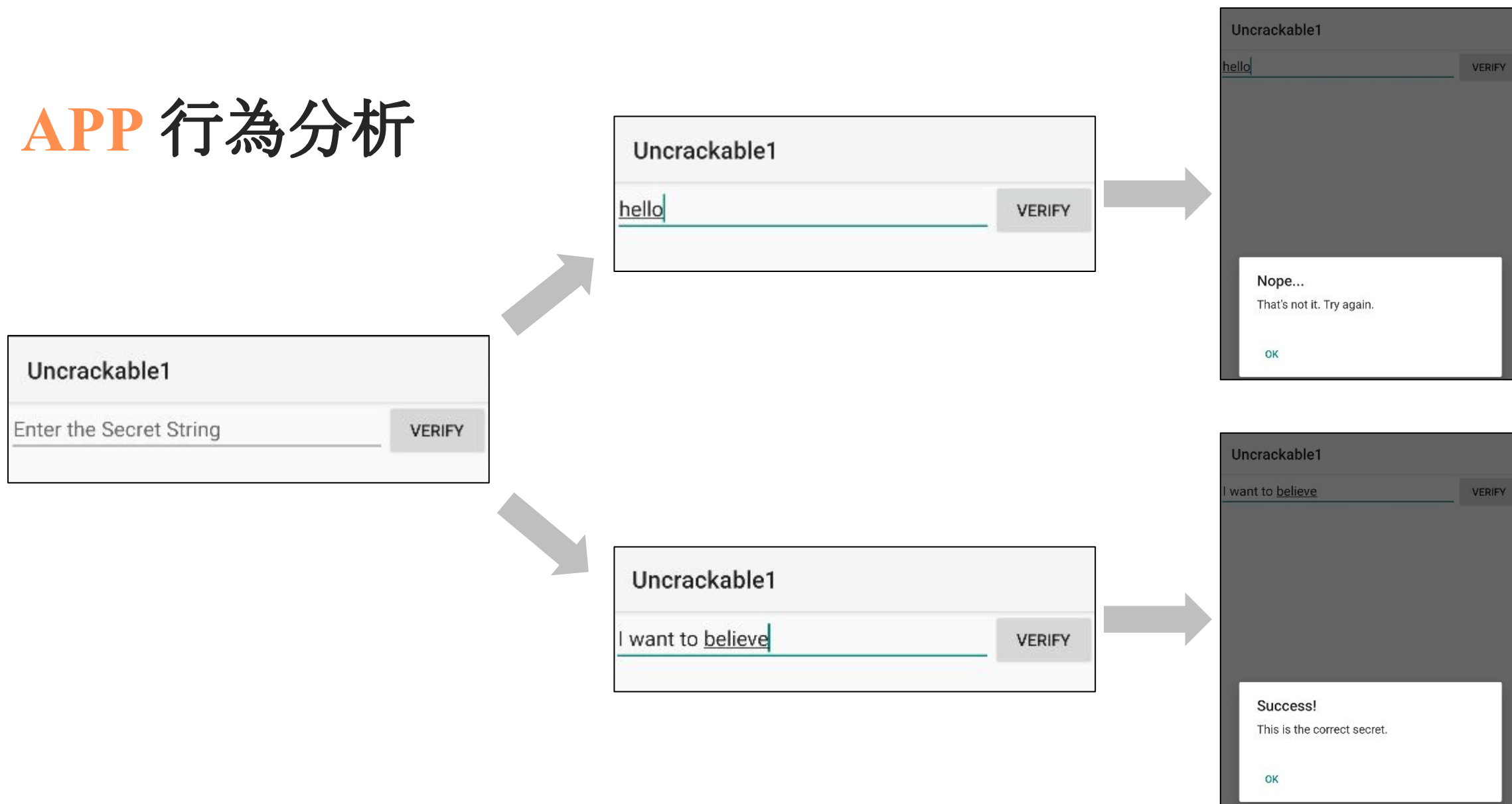


These are the UnCrackable Apps for Android and iOS, a collection of mobile reverse engineering challenges. These challenges are used as examples throughout the OWASP MASTG. Of course, you can also solve them for fun.

See <https://mas.owasp.org/crackmes> for more information.



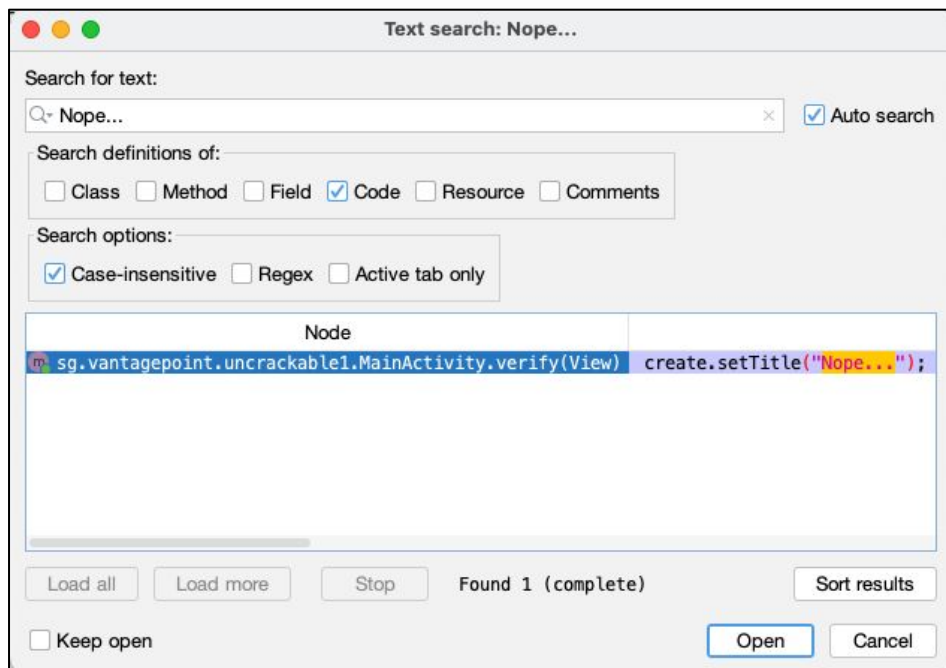
# APP 行為分析



## 逆向分析

# 字串搜尋與原始碼分析

- 在 JADX 裡搜尋錯誤訊息文字 **Nope...**, 找出驗證函數 **verify()**
- 仔細閱讀原始碼, 發現 **if(a.a(obj))** 為關鍵判斷點



```
public void verify(View view) {
    String str;
    String obj = ((EditText) findViewById(R.id.edit_text)).getText().toString();
    AlertDialog create = new AlertDialog.Builder(this).create();
    if (a.a(obj)) {
        create.setTitle("Success!");
        str = "This is the correct secret.";
    } else {
        create.setTitle("Nope...");
        str = "That's not it. Try again.";
    }
    create.setMessage(str);
    create.setButton(-3, "OK", new DialogInterface.OnClickListener() { // from c
        @Override // android.content.DialogInterface.OnClickListener
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    create.show();
}
```



<https://github.com/skylot/jadx>

## 逆向分析

# smali分析

- 將原始碼轉為 smali, 發現 `if(a.a(obj))`

對應的 smali 為 `if-eqz p1, :cond_2b`

- 使 `if-eqz p1, :cond_2b` 不成立即可顯示  
驗證成功

```
131     invoke-static {p1}, Lsg/vantagepoint/uncrackable1/a;->a(Ljava/lang/String;)Z
132
133     move-result p1
134
135     if-eqz p1, :cond_2b
136
137     const-string p1, "Success!"
138
139     invoke-virtual {v0, p1}, Landroid/app/AlertDialog;->setTitle(Ljava/lang/CharSequence;)V
140
141     const-string p1, "This is the correct secret."
142
143     :goto_27
144     invoke-virtual {v0, p1}, Landroid/app/AlertDialog;->setMessage(Ljava/lang/CharSequence;)V
145
146     goto :goto_33
147
148 :cond_2b
149     const-string p1, "Nope..."
150
151     invoke-virtual {v0, p1}, Landroid/app/AlertDialog;->setTitle(Ljava/lang/CharSequence;)V
152
153     const-string p1, "That\'s not it. Try again."
154
155     goto :goto_27
156
157 :goto_33
158     const/4 p1, -0x3
```



# 作業環境及準備工具

- MacOS
- JRE (Java Runtime Environment) or JDK  
<https://www.oracle.com/java/technologies/downloads/>
- Android Studio  
<https://developer.android.com/studio>
- smali/baksmali  
<https://bitbucket.org/JesusFreke/smali/downloads/>
- Vim 或文字編輯器
- 檔案及目錄結構

```
UnCrackable-Level1
|___ UnCrackable-Level1.apk
|___ smali-2.5.2.jar
|___ baksmali-2.5.2.jar
```

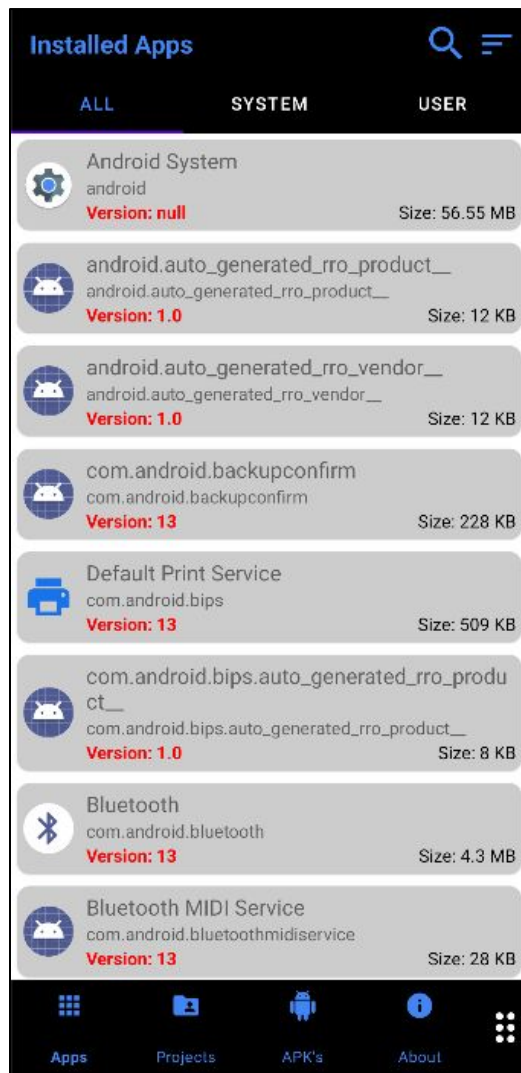
# 完整步驟

- 1 `unzip UnCrackable-Level1.apk -d UnCrackable-Level1`
- 2 `rm -r UnCrackable-Level1/META-INF`
- 3 `java -jar baksmali-2.5.2.jar disassemble UnCrackable-Level1/classes.dex -o UnCrackable-Level1/classes`
- 4 `rm UnCrackable-Level1/classes.dex`
- 5 使用 Vim 或文字編輯器開啟 `UnCrackable-Level1/classes/sg/vantagepoint/uncrackable1/MainActivity.smali`  
在 `if-eqz p1, :cond_2b` 前加上 `#` , 使該行變成註解, 修改後存檔
- 6 `java -jar smali-2.5.2.jar assemble UnCrackable-Level1/classes -o UnCrackable-Level1/classes.dex`
- 7 `rm -r UnCrackable-Level1/classes`
- 8 `cd UnCrackable-Level1; zip -r ../UnCrackable-Level1-repacked.apk *; cd ..`
- 9 `jarsigner -verbose -sigalg SHA256withRSA -digestalg SHA-256 -keystore ~/.android/debug.keystore  
UnCrackable-Level1-repacked.apk androiddebugkey -storepass android -keypass android -signedjar  
UnCrackable-Level1-repacked-signed.apk`

# DEMO



# 懶人工具: APK Explorer & Editor



- ▶ 本身也是一個開放原始碼的 Android App
- ▶ 反編譯及修改 DEX 檔
- ▶ 二次打包及重新簽章
- ▶ 專案網址: <https://github.com/apk-editor/APK-Explorer-Editor>
- ▶ 其他非開放原始碼同類工具: NP管理器、MT管理器、ApkCrack

# DEMO



# 防範措施

# 如何避免修改及二次打包

## 1. 程式碼混淆

- proGuard: <https://developer.android.com/build/shrink-code>
- BlackObfuscator: <https://github.com/CodingGay/BlackObfuscator>

## 2. 使用 C/C++ 開發 NDK 程式並搭配防逆向措施

- Android NDK 程式 (.so檔) 逆向與防逆向: <https://cyber.ithome.com.tw/2022/session-page/817>

## 3. 完整性檢查 API

- Android API: [PackageManager.requestChecksums\(\)](#)
- Google Play API: [Play Integrity API](#)

## 4. 商用 App 安全防護產品

- ICEshell: <https://iceshell.co>

# Thanks for Listening

Email: [jason@iceshell.co](mailto:jason@iceshell.co)

