Android 手機程式自動驗證系統研究

方文聘
元培科技大學
wpfang@mail.ypu.edu.tw

廖子萱
元培科技大學
qoo46122@yahoo.coom.tw

李貞穆
元培科技大學
zx7353168@yahoo.com.tw

摘要

目前手機應用程式廣泛被使用，包括 Android、i0S 平台都有提供應用程式下載使用，也因此安全性也越來越被重視，但是，Android 平台因為是開放式平台，有許多廠商都在開發軟體與硬體，多元開發的結果，會有許多不相容與安全問題出現，如何自動檢驗軟體功能與安全性，也變成重要議題，本論文提出自動驗證的方法，利用 Java 可以反組譯的特性，使用白箱測試，可以解決大量測試的問題，本論文也可以當作後續安全性與程式效率驗證的工具。

關鍵詞：APP、反組譯、智慧型手機

Abstract

Now-a-day, it is very popular to use smart phone every day. The operating systems of smart phone include Android and iOS. The corresponding app stores supply many applications to users. It is very important to protect users'; information. However, there are many compatible and security problem need to be overcome. This paper proposed a method to detect the function and security of applications automatically. Based on white box testing and decompile technology, the requirement is has been approached. The result of this paper is useful for the latter researches.

Keywords: APP, decompile, smart phone.

1. 前言

近年來，智慧型手機使用率逐年升高，根據統計有 89%的智慧型手機使用者每天都會使用到他們的手機，表 1 為智慧型手機在各年齡層的市場佔有率。

表 1、 2012 年 3 月智慧型手機在各年齡層的市場佔有率

	iPhone	Android	Microsoft	RIM
18-34 歲	43%	50%	39%	38%
35-44 歲	24%	21%	20%	25%
45-54 歲	16%	16%	21%	20%
54+ 歲	17%	13%	20%	18%

智慧型手機的功能也從通話改變為應用程式操作，使用者可以下載安裝有興趣的應用程式，但是，也因此有資訊安全尚未知的風險，以 Android 手 機為例，因為設計工具的特性與使用的習慣，可能在操作或下載時不經意開放不必要的權限，造成有心人士竊取或是操控使用者手機，比方說並透過手機連上網路把使用者的資訊複製，或者將使用者目前的地理資訊散布，造成個人資訊的洩漏。有許多案例發現目前有許多手機應用程式並不安全，舉例來說:Terry Zink[1] 提出 Android 系統的裝置利用郵件服務發送垃圾郵件，在一封發自 Yahoo!mail 伺服器的典型垃圾信件中，包含 Message-ID:1341147286.19774.androidMobile @web140302.mail.bf1.yahoo.com 垃圾郵件底部還有：“Sent from Yahoo! Mail on Android”這樣的簽名資訊。這些信都是利用 Android 設備登入信箱後發送。信件中的 IP 位址可以對發送設備進行定位，以上案例可以知道，不當下載應用程式會讓手持式裝置感染木馬程式或是病毒的可能性會更大。

因為 Android 系統的應用程式下載商店(Google Play)並不是詳細審查制，上架軟體沒有保證安全，而使用者並沒有太多方法或是能力可以檢視下載的應用程式是否會危害本身的權益。另外，目前的發展工具因為都是免費的，所以並沒有太多廠商研發如何自動檢查應用程式的程式寫作品質，所以，程式在使用中也可能有漏洞造成裝置不穩定或是讓有心人士有機會攻擊所下載的應用程式。所以本論文提出初步自動化偵測 Android 應用程式的方法，藉由 Java 程式語言的特性、逆向工程與白箱測試，可以達到軟體安全性評估與功能完善

測試的目的。

本論文其他段落包括第二段說明相關技術、第三段提出方法、第四段說明實驗結果，第五段則討論與提出未來展望。

2.相關技術說明

2.1 Android 系統簡介

Android[2]是一個以 Linux 為基礎的半開放原始碼作業系統，主要用於行動設備，由 Google 成立的 Open Handset Alliance（OHA，開放手機聯盟)持續領導與開發。架構如圖 1[3]所示。Android 作業系統使用了沙箱機制，所有的應用程式都會先被簡單地解壓縮到沙箱中進行檢查，並且將應用程式所需的權限送出給系統，並且將其所需權限以列表的形式展現出來，供使用者檢視。使用者可以根據權限來考慮是否需要安裝，使用者只有在同意了應用程式權限之後，才能進行安裝，Android 以 Linux 為核心的 Android 行動平台，使用 Java 作為開發的程式語言，Activity 類別負責建立視窗，Service 負責背景執行，透過由 ServiceConnection 和 AIDL 連結，達到多程式同時執行的效果。如果執行中的 Activity 全部畫面被其他 Activity 取代時，該 Activity 便被停止，甚至被系統清除。介面則以 View 呈現，程式人員可以透過 View 類別與「XML layout」檔將使用者介面放置在視窗上，用 xml 來設計 layout。在 Activity 中，要透過 findViewById() 來從 XML 中取得 View，View 與事件息息相關，兩者之間透過 Listener 結合在一起，每一個 View 都可以註冊 event listener。

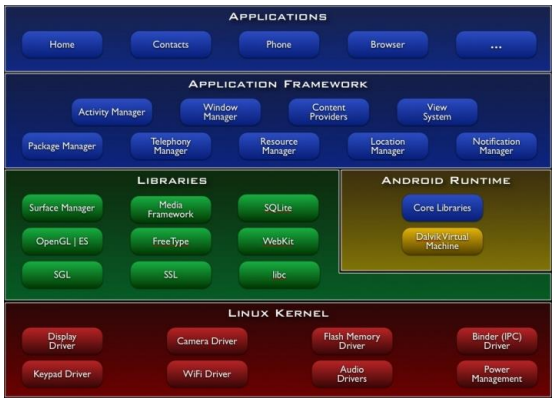


圖 1、Android 的系統架構圖

2.2 Java 語言簡介

Java[4]是一種電腦程式設計語言，擁有跨平台、物件導向、泛型程式設計的特性。 Java 程式語言的風格十分接近 C++語言。繼承了 C++語言物件導向技術的核心，Java 捨棄了 C++語言中容易引起錯誤的指標，改以參照取代，同時移除原 C++與原來運算子多載，也移除多重繼承特性，改用介面取代，增加垃圾回收器功能。在 Java SE 1.5 版本中引入了泛型程式設計、型別安全的列舉、不定長參數和自動裝/拆箱特性。Java 程式語言是個簡單、物件導向、分布式、解釋性、健壯、安全與系統無關、可移植、高效能、多執行緒和動態的語言，Java 不同於一般的編譯語言和直譯語言。它首先將原始碼編譯成位元組碼（bytecode），然後依賴各種不同平台上的虛擬機器來解釋執行位元組碼，從而實作了「一次編譯、到處執行」的跨平台特性。

2.3 AndroidManifest.xml 文件結構說明

每個 Android 的 application 都必須包含一個 AndroidManifest.xml[11]，且文件名是固定的，不能修改。應用程序需要通過它向 Android 系統提供一些必需的訊息，且需要在 application 運行前提供給系統，如下圖所示：

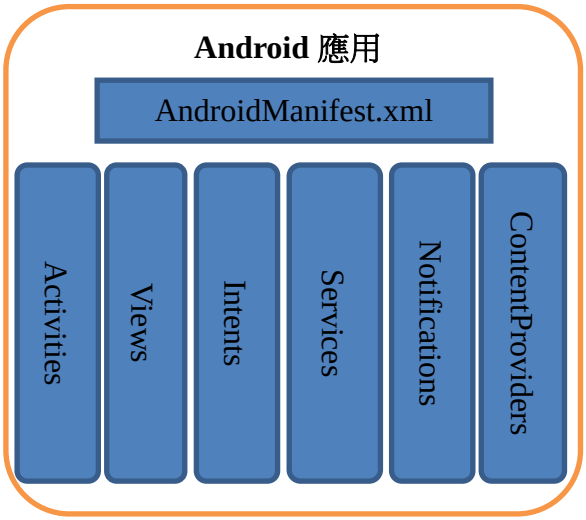


圖 2、Android 的架構圖

AndroidManifest.xml 主要包含以下功能：

1.說明 application 的 java 名稱。
2.描述 application 的 component。
3.說明 application 的 component 運行的 process。
4.設定 application 所必須具備的權限。
5.設定 application 其他的必備權限，用以 component 之間的交互。
6.列舉 application 運行時需要的環境配置訊息。

7.聲明 application 所需要的 Android API 的最低版本級別，比如 1.0，1.1，1.5。
8.列舉 application 所需要函式庫。

AndroidManifest.xml 的結構和規則：
1.元素：一個元素包含有其他子元素，必須通過子元素的屬性來設置其值，處於同一層次的元素，這些元素的說明是沒有順序的，在所有的元素中只有<manifest>和<application>是必需的，且只能出現一次。
2.屬性：除了根元素<manifest>的屬性，元素屬性的名字都是以 android:開始，所有的屬性都是可選的，但是有些屬性是必須設置的。那些真正可選的屬性，即使不存在，其也有預設的數值項說明。
3.定義類別名：所有的元素名都對應其在 SDK 中的類別名。
4.多數值項：如果某個元素有超過一個數值，這個元素必須通過重複的方式來說明其某個屬性具有多個數值項，且不能將多個數值項一次性說明在一個屬性中。
5.資源項說明：當需要引用某個資源時，其採用如下格式：@[*package*:]*type*:*name*
例如<activity android:icon="@drawable/icon" . . . >。
6.字符串值：類似於其他語言，如果字符中包含有字符"\"，則必須使用轉義字符"\\"。

2.4 反組譯工具說明

因為 Java 的特性[5]包括 1. 採用兩階段式編譯，中間碼和源碼之間非常接近、2. Java Class File PE 檔保留相當多符號資訊以及 metadata、3. 使用的指令 Opcode 不多，且重複性太高(Java 只有約 202 個 Opcode[6]，表 2 為 Android Opcode 範例)、 4. Java 的 Opcode 都相當高階、5.使用 Stack-Based 的架構，指令之間的關係簡單，容易反向推導、6. 許多程式沒有使用混淆器。7. 程式架構良好，模組切割得當，所以可以很容易的被反組譯。

本論文採用目前常見的工具 dex2jar 對 Android 應用程式進行反組譯。dex2jar 的簡單操作說明如下所示:

首先安裝工具，然後讀取檔案，取得後可以利用 jd-gui 將所有 xml 與.java 個別查看。操作流程如圖 3 所示

表 2. Dalvik_opcodes 範例

opcode	說明
const-string vx,string_id	Puts reference to a string constant identified by string_id into vx.
move-result-object vx	Move the result object reference of the previous method invocation into vx.
invoke-virtual{parameters}, methodtocall	Invokes a virtual method with parameters.

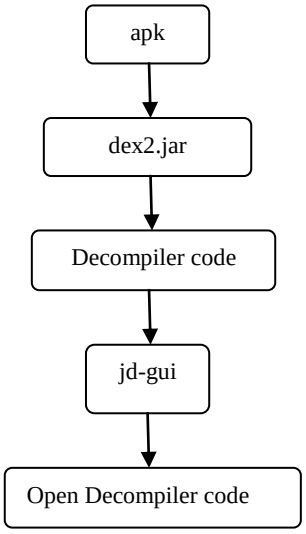


圖 3. apk Decompiler code

注意，原 apk 裡的 AndroidManifest.xml 為亂碼，經 apk-tool 反組譯後才可觀看，再透過 Android_safe.jar 確認 APK 安全性。

3. 提出方法

本論文提出自動解析 Android 應用程式的方法，所提出方法分為輸入測試、程式流程解析與程式權限解析三部分，利用現有工具發展自動化分析工具，架構如圖 4 所示。

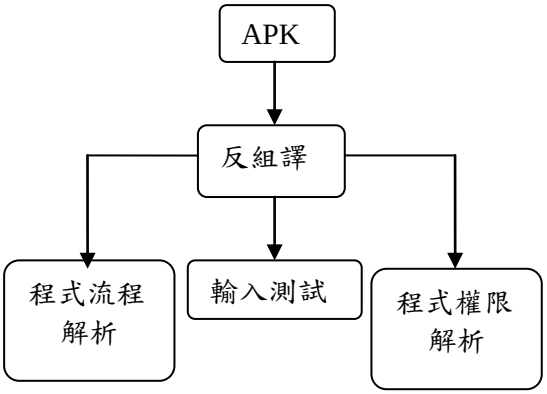


圖 4. 系統架構

3.4 輸入測試

本步驟主要是製作出自動測試軟體，作法是將程式中有關輸入的元件進行資料插入，比方說，解析後程式碼如下所示:

```
X=(TextView) findViewById(R.id.abc);
Y=X.getText();
```

將程式改為

```
X=(TextView) findViewById(R.id.abc);
Y=12;
```

此處 12 只是例子，此值由程式產生有代表性的值，包括一般性與邊界條件，再以本論文之系統自動再次編譯與執行應用程式，查詢 Log 檔看是否有異狀，如此達成自動化之目的。

3.5 程式流程解析

程式流程解析主要是要找出程式是否有不安全的寫法造成被攻擊成功的風險與因為沒有考慮到的輸入造成系統不穩定，參考文獻[7-10]，程式片段一為不安全設計風格，程式片段二為安全設計風格，可以對程式反組譯結果進行字串分析，比對，找出不良風格之程式碼。

程式片段一

```
DWORD dwret = IsAccessAllowed (….);
    if   (dwret == errot_access_denied)
      {安全性檢驗失敗
       通知使用者存取動作失敗}
    else {安全性檢驗成功
          執行這項任務}
```

程式片段二

```
DWORD dwret = IsAccessAllowed (….);
    if   (dwret == no_error)
      {安全性檢驗成功
       執行這項任務 }
    else {安全性檢驗失敗
          通知使用者存取動作失敗}
```

因為程式流程牽涉到複雜的程式架構，本論文目前針對程式分岔流程進行分析， 比方說程式判斷式如果範圍不完整即視為有潛在風險的程式，程式片段範例如下:

```
if(x>3) y=5;
if(x<3) y=6;
```

以上程式碼因為未考慮 x=3 的情形，就會在報告中提出警告。

3.6 程式權限解析

本論文對程式權限解析主要是先解析反組譯後的 XML 檔，流程如圖 5 所示，將有開放相關權限找出，可能有風險的權限及說明如表 2 所示。

實際流程為先利用 dex2jar 將 APK 反組譯，再透過 jd-gui 開啟反組譯過後的程式碼，讀取 AndroidManifest.xml ，搜尋該文字檔的所有權限相關設定(permission)，再讀取所有將所有 java 類別內的程式碼，對照所有使用相關權限的指令，與該程式說明進行比對，判斷該 APK 是否有加入不該存權限在的功能，以判別 APK 是否內藏「惡意程式」。如發現到 APK 有惡意程式便紀錄於分析的 Log 檔中，本論提出方法也可進行刪除該不當權限之相關程式碼。

比對方法如圖 6 所示，如果是不當的權限設定，便於檢測報告中提出警告。

舉一例說明，若 AndroidManifest.xm 檔內容有以下片段

```
<uses-permission android:name = "android.permission.INTERNET"></uses-permission>
```

但是，如果程式名稱或是程式碼中無網路連線功能之相關指令文字出現於程式碼中，或該應用程式說明並沒有提到 internet 的相關功能，此時即在檢測檔中警告使用者或是開發人員，程式邏輯有潛在風險。

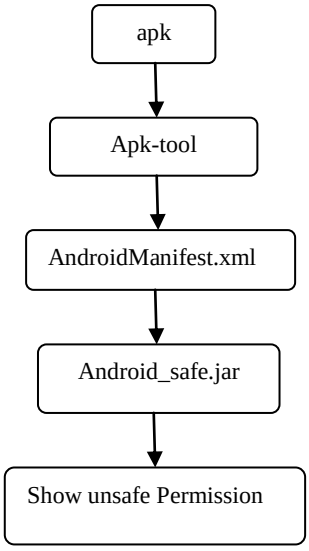


圖 5. Decompiler AndroidManffest

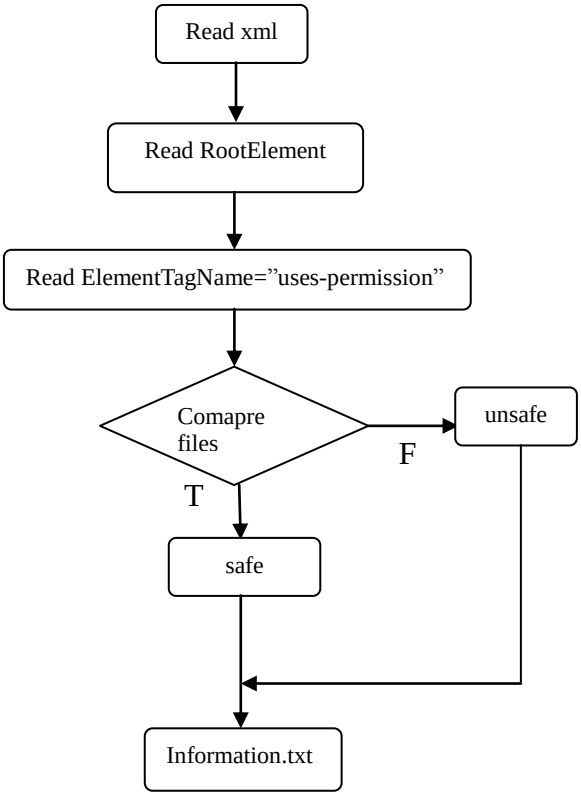


圖 6. 權限解析流程圖

表 3. 可能有風險的權限

ACCESS_COARSE_LOCATION 允許取得使用者使用基地台位置權限
ACCESS_FINE_LOCATION 允許取得使用者 GPS 位置權限
BRICK 允許取得可以關掉裝置權限
CALL_PHONE 允許不經同意撥號權限
CALL_PRIVILEGED 允許不經同意撥所有號碼權限
CAMERA 允許使用照相機
PROCESS_OUTGOING_CALLS 允許監控打電話權限
READ_CALENDAR 允許讀取行事曆權限
READ_CONTACTS 允許讀取聯絡資料權限
READ_PHONE_STATE 允許讀取唯讀電話狀態權限
READ_SMS 允許讀取唯讀簡訊權限
RECEIVE_BOOT_COMPLETED 允許收到 ACTION_BOOT_COMPLETED 訊息權限
SEND_SMS 允許傳送簡訊權限
WRITE_SECURE_SETTINGS 允許讀寫系統安全設定權限
WRITE_SMS 允許輸入簡訊權限
WRITE_EXTERNAL_STORAGE 允許寫入外部儲存權限
WRITE_SETTINGS 允許讀取系統設定權限

4. 實驗結果

本論文提出對手持式系統反組譯以便可以自動分析程式的方法，在說明實驗結果前，先對反組譯實驗進行說明，結果如下所示:表 4 為元件編號反組譯前後對照表、表 5 為變數宣告反組譯前後對照表、表 6 為主程式反組譯前後對照表、表 7 為輸入程式碼反組譯前後對照表、表 8 為迴圈反組譯前後對照表、圖 7 則展示修改前後編譯執行結果、圖 8 則展示系統執行結果。

表 4. 元件編號比較表

原始檔案內容	public static final class id {
	public static final int button1=0x7f070000;
	public static final int button2=0x7f070003;
	public static final int editText1=0x7f070001;
	public static final int editText2=0x7f070004;
	public static final int menu_settings=0x7f070006;
	public static final int textView1=0x7f070002;
	public static final int textView2=0x7f070005;
	}

Apk檔反組譯後內容	public static final class id
	{
	public static final int button1 = 2131165184;
	public static final int button2 = 2131165187;
	public static final int editText1 = 2131165185;
	public static final int editText2 = 2131165188;
	public static final int menu_settings = 2131165190;
	public static final int textView1 = 2131165186;
	public static final int textView2 = 2131165189;
	}

表 5. 變數宣告編譯前後對照表

原始檔案內容	Button bt , bt1;
	EditText et , et1;
	TextView tv , tv1;
	int ii;
	int iii=32342;
	String ssss;
	String ss2323="";
	String sse8="ghfddf";
	int a1=1,a2=2,a3=3,a4=4,a5=a1+a2;
	int a6=a3+a5;

Apk檔反組譯後	int a1 = 1;
	int a2 = 2;
	int a3 = 3;
	int a4 = 4;
	int a5 = this.a1 + this.a2;
	int a6 = this.a3 + this.a5;
	Button bt;

內容	Button bt1; EditText et; EditText et1; int ii; int iii = 32342; String ss2323 = ""; String sse8 = "ghfddf"; String ssss; TextView tv; TextView tv1;

表 6. 主程式編譯前後對照表

原始檔案內容	protected void onCreate(Bundle savedInstanceState)
	{
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_t1);
	int b1=1,b2=2,b3=3,b4=4,b5,b6;
	b5=(b1*b4+b2)/b3;
	findViews();
	}

Apk檔反組譯後內容	protected void onCreate(Bundle paramBundle)
	{
	super.onCreate(paramBundle);
	setContentView(2130903040);
	(6 / 3);
	findViews();
	}

表 7. 為輸入程式碼編譯前後對照表

原始檔案內容	tv=(TextView)findViewById(R.id.textView1);
	et=(EditText)findViewById(R.id.editText1);
	bt=(Button)findViewById(R.id.button1);
	bt1=(Button)findViewById(R.id.button2);
	et1=(EditText)findViewById(R.id.editText2);
	tv1=(TextView)findViewById(R.id.textView2);
	String s1="a",s2="c",s3,s4="",s5="asd",s6;
	String ss="";
	ss+=s1+s2;
	int a=1 , b=2 , c=3 , d=4 , e , f;
	e=a+b+c+10+d;
	f=5+3-4*2/(4-2)+c;

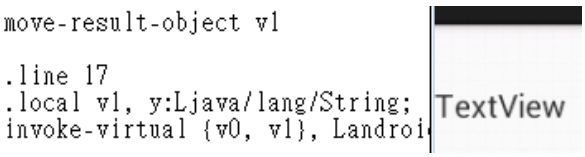
Apk	this.tv = ((TextView)findViewById(2131165186));

檔反組譯後內容	this.et = ((EditText)findViewById(2131165185));
	this.bt = ((Button)findViewById(2131165184));
	this.bt1 = ((Button)findViewById(2131165187));
	this.et1 = ((EditText)findViewById(2131165188));
	this.tv1 = ((TextView)findViewById(2131165189));
	new StringBuilder(String.valueOf(""))
	.append("a").append("c").toString();
	(4 + 16);
	(3 + 4);

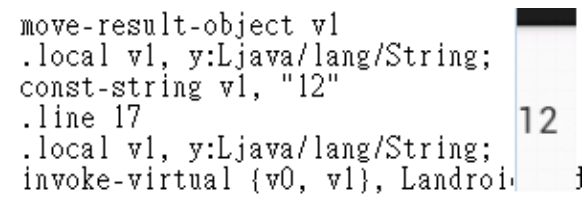
表 8. 迴圈編譯前後對照表

原始檔案內容	for(int i = 1 ; i <= n ; i++)
	sum+=i;
	tv1.setText(String.valueOf(sum));

Apk檔反組譯後內容	int i = 0;
	String str = MainActivity.this.et1.getText().toString();
	for (int j = 1; ; ++j)
	{
	if (j > Integer.valueOf(str).intValue())
	{
	MainActivity.this.tv1.setText(String.valueOf(i));
	return;
	}
	i += j;
	}

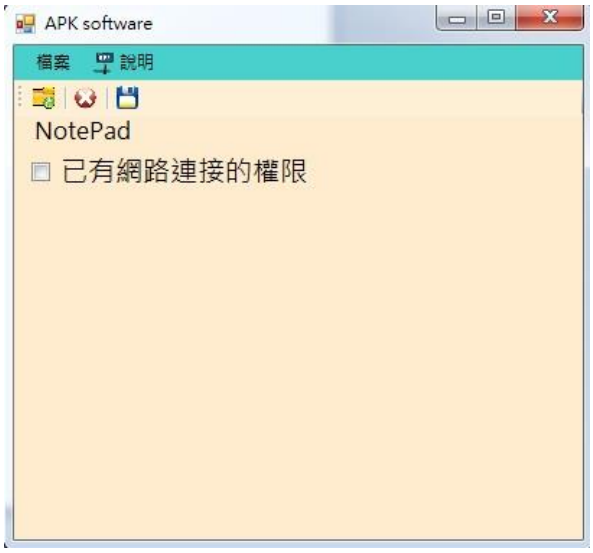


(a)

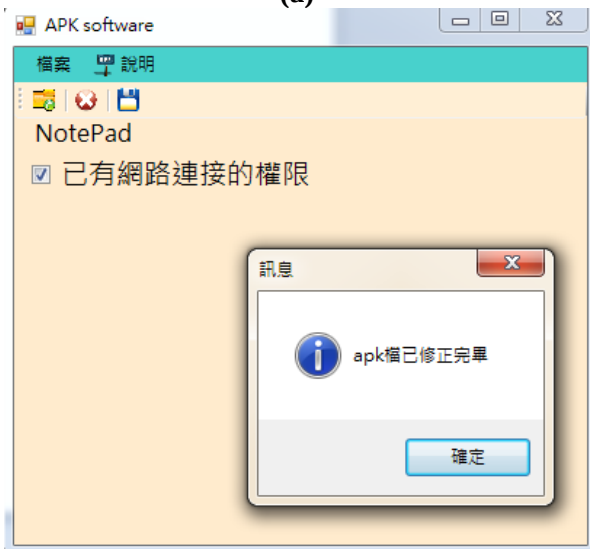


(b)

圖 7 實驗結果例子 (a) v1 值未修改，輸出為 "TextView" (b) v1 修改為"12"，輸出為"12"



(a)



(b)

圖 8 系統執行結果 (a)引入檔案並 SHOW 出所有權限
(b)選擇要刪除之權限並執行

5. 結論與未來展望

由於 Android 的權限必須宣告在 AndroidManifest.xml 底下，否則無法執行此權限之應用。本論文透過反組譯取得 AndroidManifest.xml 並解析此檔已獲得此 apk 的權限，並刪除不當之權限來達到封鎖惡意程式執行的可能性。

本論文研究如何自動檢驗 Android 應用程式的正確性的方法，並且提出安全性檢查的方法，可以解決目前許多手持式裝置開發公司所面對大量測試不同機種不便的困境的方向，基於本研究，未來可以做到自動檢測程式風格、建議使用者是否應該下載與避免有漏洞或是減少被惡意傳布的應用程式。

參考文獻

[1] http://blogs.msdn.com/b/tzink/archive/2012/07/03/spam-from-an-android-botnet.aspx。

[2] http://zh.wikipedia.org/wiki/Android

[3] 陳鍾誠 (2010 年 09 月 15 日)，Android 的系統架構，陳鍾誠的網站，取自 http://ccckmit.wikidot.com/ga:architecture ，網頁修改第 4 版。

[4] http://zh.wikipedia.org/wiki/Java

[5] http://www.ithome.com.tw/itadm/article.php?c=32297

[6] "Dalvikopcodes," http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

[7] M. Howard and D. LeBlanc, *Writing secure code* 2nd ed., Microsoft Press, 2002. http://www.microsoft.com/MSPress/books/5957.asp

[8] R.C. Seacord, Secure Coding in C and C++ , Addison-Wesley, 2005. http://www.cert.org/books/secure-coding/

[9] M.G. Graff and K.R. van Wyk, *Secure Coding: Principles and Practices*, O'Reilly, 2003.

[10] J. Viega and M. Messier, *Secure Programming Cookbook*, O'Reilly, 2003.

[11] http://www.moandroid.com/?p=80

[12] 羅日宏，Android 上的入侵偵測系統，2010，取自 http://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/login?o=dnclcdr&s=id=%22098NCTU5394097%22.&searchmode=basic

[13] 吳政隆，以 XML 為資料擷取界面之審計系統實作，2002，取自 http://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/login?o=dnclcdr&s=id=%22090CYCU5385018%22.&searchmode=basic

[14] 李逸群，網頁異動偵測技術在網際網路新聞資訊節取上之應用，2004，取自 http://thesis.lib.ncu.edu.tw/ETD-db/ETD-search-c/view_etd?URN=91423017

[15] 薛肇仁，網頁資訊擷取系統－ 資料轉換及排程啟動，2012，取自 https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CEQQFjAD&url=http%3A%2F%2Fethesys.library.ttu.edu.tw%2FETD-db%2FETD-search%2Fgetfile%3FURN%3Detd-1110112-003335%26filename%3Detd-1110112-003335.pdf&ei=B3kIUceTM8y6lQXOg4C4BQ&usg=AFQjCNFhNB-ifh35UM75aFGhUYyNuKNyKA&sig2=gXaNlVWEm3oWJoNnQvYPZg&bvm=bv.41642243,d.dGI&cad=rja

[16] John Cox, Are smartphone viruses really a threat to your network?, June 23, 2008. http://www.networkworld.com/news/2008/062308-wireless-questions-1.html