Android 應用程式之資訊安全靜態檢測
A Study of Static Analysis for Android Applications Security Detection

郭忠義、林宛瑩
國立臺北科技大學資訊工程系
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Email: { jykuo, t102598034}@ntut.edu.tw

摘　要

隨著智慧型手機的普及化，各式各樣的手機應用程式層出不窮，其中涉及的資訊安全問題卻時常被開發者、使用者所忽略，導致各種問題的發生。為解決行動應用的資訊安全問題，現有許多的資訊安全分析工具，但這些工具缺少一個統一的評測標準，各家分析結果各有偏頗，使用者針對不同面向的資訊安全問題可能需要使用多個分析平台、分析工具才能逐步找出安全漏洞。為改善此情況，本研究以經濟部工業局所訂定的行動裝置資訊安全檢測基準做為開發基礎，整合應用現有的幾項程式測試工具，分析各自的優、缺點，來對應用程式的發布安全、使用者的敏感資料、以及程式碼安全等問題進行掃描分析，找出應用程式本身設計中所包含的問題與漏洞。

關鍵字： Android 應用程式、安全性檢測、靜態分析

Keywords ： Android Application, Security Detection, Static Analysis

一、 緒論

面對市面上五花八門的手機應用程式，資訊安全的議題愈發的受到重視，惡意行為通常都發生在看不到的程式背後，一般的使用者在下載和使用應用程式時，很難直接判斷一個應用程式是否安全，也因此，陸續有許多的測試工具、測試平台上線，提供給使用者和開發者來檢測應用程式，如常見的VirusTotal，是一個集合眾多防毒軟體引擎的線上服務，能夠快速的檢查應用程式是否包含病毒，及提供檔案相關資訊。

除了惡意軟體之外，應用程式本身的設計問題也可能包含資訊安全上的漏洞，前宏碁資安工程師於 2014 年針對多款政府應用程式進行分析，發現多個應用程式皆含有資訊安全弱點，雖然不一定會造成個資外洩，但仍然是需要重視的開發問題。經濟部工業局亦於 2014 年規劃制定資訊安全檢測標準，來提升國內應用程式之基本防護能力，本研究即以此作為開發基礎，除了對應用程式有更全面的檢測之外，亦涵蓋當前測試平台已有的測試項目內容，以期擁有更好的檢測能力。

本文的章節組織如下，第二章將探討相關文獻，第三章介紹本論文之設計理論及使用的開發工具，第四章將以第三章所提出的設計方法進行實作開發，第五章則為最後的結論。

二、 相關文獻探討

2.1 Android Permission

Android 應用程式若要使用外部資源，都需要在 AndroidManifest.xml 檔案中進行註冊，早期Kirin[1]是一個較常用的檢測工具，Kirin 透過特定危險權限的組合來定義規則，以此判斷權限的危險性，但缺點是誤報率較高。

Felt 等人認為 Android 權限機制常存在溢權(Overprivilege)問題[2]，應用程式若申請過多的權限，將使得應用程式存在遭受提升權限攻擊(Privilege Escalation)[3]、Intent 劫持等惡意行為的安全隱患；Kathy 等人則進一步對 Android 權限進行分析[4]，發現在被記錄的情況下，約有 22%的權限是不必要的。

2.2 Android Static Analysis Tools

近年來，Android 安全相關的研究日益增加，相關的檢測工具也陸續上市，其中一種知名的靜態分析工具是 Androguard[5]，它提供許多 python 的分析模組，來對應用程式做反向工程和檢測，現今有許多檢測框架皆是延伸自 Androguard，如 Androwarn [6]，結合 Chilkat、Jinja 等框架，掃描應用程式的系統資訊、認證憑證、是否曝露地理位置、PIM 訊息等等，再輸出 txt 文字檔或 html 網頁給使用者；AndroBugs[7]，除了檢查程式是否包含危險 command 之外，亦結合 MongoDB 來提供大量分析應用程式的功能，再從中收集資訊。

另外一種分析方式是偵測動態分析之後的結果，如資料流分析工具 FlowDroid[8]，它基於 Java 分析工具 Soot，針對 Android 的生命週期進行汙點分析，並提供函數關係圖，以提高分析的準確率。

惡意程式檢測也是靜態分析中的一大議題，常見的檢測方式有權限分析、惡意程式特徵分析、ByteCode 分析等等，Faruki 等人介紹的 AndroSimilar 檢測工具[9]，即是透過統計相似度及檢測惡意程式的簽證，來避免程式碼的混淆(Obfuscation)和重新封裝(Repackaging)。

現今有越來越多的平台結合上述幾種分析方式，提供整合性的測試，如 DroidAnalyst[10]，是一

個包含動態(尚未上線)和靜態分析的線上分析平台，其靜態分析部分包括對權限的評估、使用的檔案、元件互動圖等等功能，也提供惡意程式分析平台 AndroSimilar 和 VirusTotal 的掃描結果，並整合分析結果報表給使用者。

2.3 OWASP Top 10

OWASP(The Open Web Application Security Project, 開放網路軟體安全計畫)**錯誤！找不到參照來源。**是一個資訊安全非營利組織，其研議的軟體安全標準是許多專家、網站所遵循的安全標準，而其中最知名的計畫 OWASP Top 10[12]，是針對常見的網路應用系統安全弱點提供相關的定義及防護建議，網站開發者在開發上建議能夠參照其所提出的標準，避免存在容易遭受攻擊的安全漏洞。另一方面，OWASP 除了對網頁應用程式提出 Top 10 之外，在行動應用方面亦公布相關的資訊安全問題—Top 10 Mobile Risks[13]。

OWASP Mobile Top 10 列舉幾項常見的行動應用資訊安全問題，例如伺服器端的安全控制、不當的使用 Intent、系統權限、不安全的儲存造成資料洩漏、不安全的資料傳輸、加密演算法不足、程式碼遭到竄改等等，行動應用開發者應該對這些資訊安全弱點有所了解，減少行動應用程式設計上的缺陷。

三、 靜態安全檢測設計

經濟部工業局所訂定之資訊安全評測標準，各項目內容中涵蓋幾個資訊安全的常見範疇，而本研究將分別針對當中幾個項目，提出分析方式及使用工具，以滿足其訂定之檢測規範。

3.1 經濟部工業局資安評測標準

為協助行動應用程式開發者提升應用程式開發品質，經濟部工業局依據行動應用 App 基本資安規範[14]，並參考相關國際資訊安全規範，如 OWASP Mobile Top 10[13]、NIST(National Institute of Standards and Technology)的 Special Publication 800-163 Vetting the Security of Mobile Applications[15]等，訂定相關基本資訊安全檢測項目[16]，分為五大面向：行動應用程式發布安全、敏感性資料保護、付費資源控管安全、行動應用程式使用者身分認證、授權與連線管理安全、及行動應用程式碼安全。

3.1.1 行動應用程式發布安全檢測

市面上現有的 Android 行動應用程式發布平台，除了官方的 Google Play 之外，Amazon、微軟、遠傳、三星等各家軟硬體廠商也有各自的發布平台。行動應用程式發布安全檢測中針對 App 之發布、更新、問題回報等方面之資訊安全進行檢測，其中，本研究對 App 之發布部分進行檢測設計。

對於發布平台的選擇，本研究排除純遊戲發布端及電信業者自有平台，篩選發布資訊完整，且內容包含權限用途宣告者，選擇以 Google Play 作為可信任來源之應用程式發布資訊的蒐集來源，再與應用程式內所宣告的權限進行資料分析比對，以檢測應用程式是否發布於行動應用程式商店，並且對於所使用之權限做出適當之使用說明。

Android 的權限管理亦是重要的 Android 安全機制之一。本研究以 AndroidUnusedPermissions[17]，一個根據 Kathy 等人對 Android 權限所做的規範[4]，所開發的掃描工具，分析應用程式中未使用到的權限，提供非必要的權限列表給開發者做參考。但根據實驗觀察，AndroidUnusedPermissions 所檢測之結果包含若干誤差的情況，因此本研究進一步對產出之結果檔案進行二次分析，確認應用程式是否有使用該權限功能。

3.1.2 敏感性資料保護

敏感性資料泛指使用者操作應用程式時建立或儲存於行動裝置之資訊，例如個人資料、密碼、金鑰、即時通訊訊息、通訊錄、地理位置、通話紀錄等等，這些訊息的洩漏可能對使用者造成損害，對於敏感性資料與個人資料相關訊安全之檢測，包括敏感性資料的蒐集、利用、儲存、傳輸、分享及刪除等。

MobSF(Mobile Security Framework)[18] 是由 Ajin Abraham 於 2015 年所開發的一款開源安全測試框架，提供對 Android 及 iOS 的靜態檢測及動態監測的功能，它能掃描 APK 文件的原始碼，收集相關訊息，以便於識別程式的具體漏洞，如 XXE(XML External Entity)、SSRF(Shanghai Synchrotron Radiation Facility)、IDOR(Insecure Direct Object References)、Path Traversal 等等。

敏感性資料的蒐集、利用與儲存部分，本研究藉由 MobSF 框架來掃描程式的關鍵字及呼叫的方法(Method)，找出不安全的資料使用，和曝露的(Exported)可能被外部呼叫的公開元件等。

資料分享方面，不同程式之間對資料的分享，Android 透過 Intent 來實現不同頁面(Activity)之間的溝通，而 Intent 又可藉由啟動 Activity、Service 和 Broadcast 方法呼叫，因此本研究蒐集 Intent 呼叫資訊及外部 URI 字串等，避免不恰當的資料分享行為。

中間人攻擊(Man-in-the-Middle, MITM)是常見的惡意攻擊方法之一，攻擊者在通訊中的兩端之間加入一個惡意的節點，以此攔截通訊內容。Android 通過 SSL/TLS 加密 API 來傳輸訊息，德國兩所大學研究人員，Fahl 等人[19]針對 13500 個應用程式進行分析，發現約 8%的應用程式含有中間人攻擊的安全漏洞，因此設計一套分析軟體 MalloDroid[20]，延伸自 Androguard，針對應用程式的 http/https 請求，檢核其 SSL 憑證的有效性。本研究利用 MalloDroid 對有使用網路傳輸的應用程式，偵測是否包含 SSL/TLS 傳輸漏洞，以此達

到資料傳輸之檢測。

3.1.3 付費資源控管安全

付費資源控管安全的檢測包括付費資源之使用與控管。Google Play 對於程式內金流機制 (In-App Billing, IAB)提供兩種方式可供測試，一種是 Testing with Static Response，透過特定 Product ID 模擬送出成功購買商品、取消購買商品、商品退費、購買未上架商品等狀況的交易請求，以供開發者測試對應之處理流程；第二種是 Setting Up for Test Purchases，開發者可以設定不會進行實際扣款的測試人員帳號，藉此進行 IAB 的測試。

對於已上架之應用程式的付費資源控管測試，由於無法取得開發者設定的測試帳號，Google Play 也無額外提供的統計 API，因此僅能對付費資源聲明做偵測，本研究將此涵蓋於發布檢測中。

3.1.4 身分認證、授權與連線管理安全

針對行動應用程式身分認證、授權與連線管理安全之相關資訊安全檢測，包括使用者身分認證與授權及連線管理機制等。由於身分認證機制各應用程式的架構皆不同，尚無通用檢測方法，較難自動檢測或檢測結果不一致，而連線安全之管理則依靠動態檢測，因此本研究僅對伺服器憑證驗證之相關資訊進行確認，如憑證之有效期間、發行單位等。

3.1.5 行動應用程式碼安全

行動應用程式開發之相關資訊安全內容包括防範惡意程式碼與避免資訊安全漏洞、行動應用程式完整性、函式庫引用安全與使用者輸入驗證等。應用程式若在設計中有缺陷，容易使系統在未經使用者的同意下接受惡意指令，侵害使用者權益，常見的惡意行為包含注入攻擊(Injection)、跨站腳本攻擊(Cross-site scripting, XSS)等等，在資訊安全檢測基準中，列舉 SQL Injection、JavaScript Injection、Command Injection 、Local File Inclusion 、XML Injection、Format String Injection、Intent Injection 等檢測項目。

PMD[21]是一個 Java 的靜態檢測工具，它原本是設計來對原始碼結構進行分析，找出可能的 Bug、無法執行的程式碼、不良的程式碼、複雜或重複的程式碼等等設計缺陷，但其 Rule-base 的特性讓開發者擁有更多的擴充性。本研究將基於 Gotham Digital Science(GDS)所提供的 PMD Secure Coding Ruleset[22]來對反編譯後的應用程式原始碼進行掃描，找出 OWASP 所定義的資訊安全弱點。但 GDS PMD Secure Coding Ruleset 主要是針對 OWASP Top 10 中的五項弱點：Injection、XSS、Insecure Cryptographic Storage、Failure to Restrict URL Access、Unvalidated Redirects and Forwards 進行掃描，對於行動應用程式來說可能有所不足，因此本研究將進一步參考 OWASP Mobile Top 10 中的內容，及資訊安全檢測基準所列的項目來設計檢測規則，擴充 PMD 的應用範圍。

四、 靜態安全檢測系統實作

根據經濟部工業局的行動應用資訊安全檢測標準，將第三章所提出的各檢測章節內容重點，設計系統流程及程式架構，分別進行實作開發。

4.1 系統架構與流程

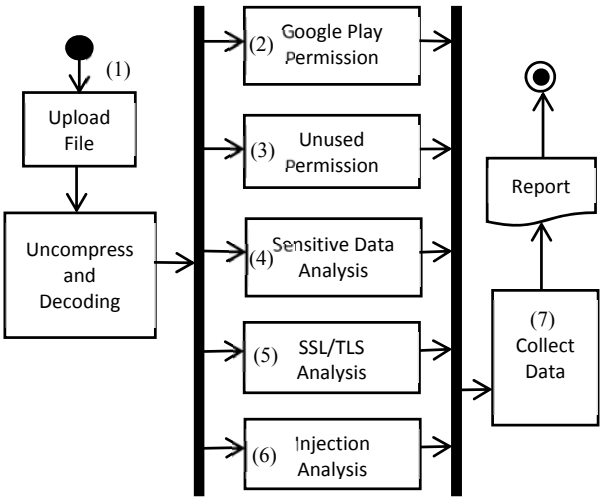


圖1. 系統流程圖

(1) 上傳應用程式後，對上傳的 APK 檔案進行解壓縮及反編譯動作，得到 APK 的原始碼。
(2) 判斷 APK 是否發布於 Google Play，若是，則蒐集其發布資訊，與 AndroidManifest.xml 所宣告的權限進行比對分析。
(3) 透過 AndroidUnusedPermissions 分析非必要之 Android 權限，再對結果所列之檔案進行掃描，進一步降低錯誤率。
(4) 敏感性資料保護部分使用 MobSF 框架，蒐集應用程式憑證資訊、Code Analysis 中對程式碼之掃描結果及 Manifest Analysis 中對 Manifest 掃描後找出的弱點結果。
(5)若應用程式有使用網路，則利用 MalloDroid 分析是否包含傳輸漏洞。
(6) 使用 PMD 及 GDS Security Coding Ruleset 分析應用程式程式碼，檢測其是否包含 OWASP 等資訊安全漏洞。
(7) 整合各分析結果，產出檢測報表。

4.2 系統實作

4.2.1 發布安全

本研究以 Google Play 作為可信任之應用程式發布商店，收集發布來源中對各項權限的宣告內容，和應用程式所宣告的權限做比對，以此判斷應用程式發布時是否有對使用的權限做出完整使用宣告。

應用程式開發者在開發應用程式時需要對要使用的權限在 AndroidManifest.xml 文件中做出宣

告，才能在應用程式中使用該項功能，而 Android 開發者網站中所列權限列表有 135 項[23]，本研究另外統計 28 項權限，共 163 項一般 Android 應用程式中會宣告的權限項目(不包含使用者自訂權限)。

```
<uses-permission android:name="android.permission.CAMERA">
<uses-permission android:name="android.permission.VIBRATE">
<uses-permission android:name="android.permission.READ_CONTACTS">
<uses-permission android:name="android.permission.INTERNET">
<uses-permission android:name="android.permission.READ_PHONE_STATE">
<uses-permission android:name="android.permission.CALL_PHONE">
```

圖2. Android Permission 宣告範例

Google Play 的使用權限宣告項目官方並未提供完整的權限宣告定義資訊，本研究根據 Google Play 熱門應用程式，整理常見的 150 項權限宣告內容，即檢測的比對對象，判斷應用程式是否有未宣告之發布資訊。

圖3. Google Play 權限使用宣告範例

根據以上兩項資料蒐集結果，比對應用程式中所宣告的權限及 Google Play 中所宣告的權限，判斷是否有未聲明使用的權限，執行結果如圖 4。

Android Manifest權限列表

#	Permission	Description
一般權限		
1	android.permission.CAMERA	Required to be able to access the camera device.
2	android.permission.VIBRATE	Allows access to the vibrator
3	android.permission.READ_CONTACTS	Allows an application to read the user's contacts data.
4	android.permission.INTERNET	Allows applications to open network sockets.
5	android.permission.READ_PHONE_STATE	Allows read only access to phone state.
6	android.permission.CALL_PHONE	Allows an application to initiate a phone call without going throug
7	android.permission.ACCESS_COARSE_LOCATION	Allows an app to access approximate location derived from netw
8	android.permission.ACCESS_FINE_LOCATION	Allows an app to access precise location from location sources
9	android.permission.WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.
10	android.permission.ACCESS_NETWORK_STATE	Allows applications to access information about networks
11	android.permission.ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks
12	android.permission.RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLET
13	android.permission.RESTART_PACKAGES	This constant was deprecated in API level 8. The restartPackage
14	android.permission.RECEIVE_SMS	Allows an application to monitor incoming SMS messages, to re

圖4. 發布資訊檢測報表

而本研究另外對於應用程式中宣告的權限進行分析，檢測是否宣告過多權限，造成溢權問題。執行結果如圖 5。

未使用權限列表

列出非必要之權限(Analysis by AndroidUnusedPermission, which according PSCount.)

#	Permission Item
1	android.permission.ACCESS_FINE_LOCATION
2	android.permission.READ_CALENDAR
3	android.permission.CHANGE_WIFI_MULTICAST_STATE
4	android.permission.RECORD_AUDIO
5	android.permission.READ_CONTACTS
6	android.permission.MODIFY_AUDIO_SETTINGS
7	android.permission.WRITE_EXTERNAL_STORAGE
8	android.permission.BLUETOOTH_ADMIN
9	android.permission.WRITE_CALENDAR

圖5. 溢權檢測報表

4.2.2 敏感性資料保護

大部分的檢測平台皆包含掃描資料使用的能力，例如使用到的檔案、使用的 API、宣告的 Activities、外部 URL 等等資訊，本研究亦涵蓋這些資訊，除了將這些檢測內容依據使用、傳輸、共享、資料庫等進行分類便於閱讀之外，也提供檔案的連結能夠快速連結至原始碼檢視頁面，查閱有問題的程式碼。圖 6 為部分之敏感性資料檢測報表畫面，圖 7 為資料庫安全檢測中有問題之程式碼。

資料利用

Sensitive Informations
Files may contain hardcoded sensitive informations like usernames, passwords, keys etc.
Globals.java ZaAuthActivity.java

Application Directory
App can write to App Directory. Sensitive Information should be encrypted.
None

External Storage
App can read/write to External Storage. Any App can read data written to External Storage.
BackupRestore.java ZapService.java Logger.java WrkMalware.java WrkFileBrowser.java

Temporary File
App creates temporary file. Sensitive information should never be written into a temp file.
f.java

Android Secret Code
None

圖6. 敏感性資料檢測報表

Back com/mobileapptracker/f.java

```
1
2
3  package com.mobileapptracker;
4
5  import android.content.Context;
6  import android.database.sqlite.SQLiteDatabase;
7  import android.database.sqlite.SQLiteOpenHelper;
8  import android.util.Log;
9
10 final class f
11   extends SQLiteOpenHelper
12 {
13   f(Context paramContext)
14   {
15     super(paramContext, "MAT", null, 1);
16   }
17
18   public final void onCreate(SQLiteDatabase paramSQLiteDatabase)
19   {
20     paramSQLiteDatabase.execSQL("create table referrer_apps (_id integer
21   }
22
23   public final void onUpgrade(SQLiteDatabase paramSQLiteDatabase, int para
24   {
25     Log.w("Content provider database", "Upgrading database from version "
26     paramSQLiteDatabase.execSQL("DROP TABLE IF EXISTS siteids");
27     onCreate(paramSQLiteDatabase);
28   }
29 }
```

圖7. 資料庫安全有漏洞之程式碼

而在各項檢測中，資料傳輸檢測部分還包含 SSL/TLS 弱點分析，若程式碼包含 HTTP/HTTPS 的漏洞，報表中一樣會列出有問題的檔案及對應的外部連結，報表結果如圖 8。

SSL/TLS Vulnerability

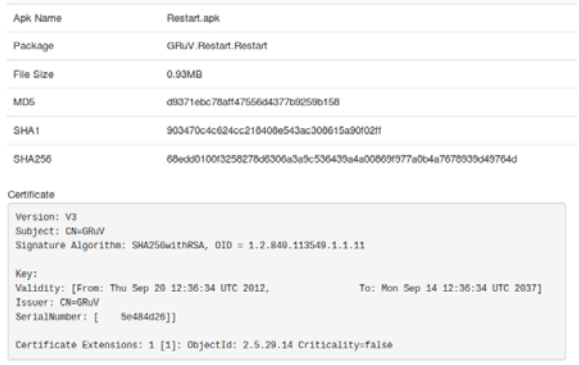
Broken SSL certificate validation in Android Apps. But there may be deviate in detection.

File	External Reference
Insecure SSL Socket	
Globals$2	trustOnlyPinned
HttpConnection$2	trustOnlyPinned
Hostname Verifiers	
Globals$1	trustOnlyPinned
HttpConnection$1	trustOnlyPinned
Received SSL Errors	

圖8. SSL/TLS 漏洞檢測範例

4.2.3 授權管理

授權管理部分本研究蒐集應用程式之授權憑證資訊，包含發行單位、有效期間、加密演算法等，將這些資訊列於應用程式資訊中以報表顯示，如圖 9 所示。



Apk Name	Restart.apk
Package	GRuV.Restart.Restart
File Size	0.93MB
MD5	d9371ebc78aff47556d4377b9259b158
SHA1	903470c4c624cc218408e543ac308615a90f02ff
SHA256	68edd0100f3258278d6306a3a9c536439a4a00669f977a0b4a7678939d49764d

Certificate

```
Version: V3
Subject: CN=GRuV
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

Key:
Validity: [From: Thu Sep 20 12:36:34 UTC 2012,              To: Mon Sep 14 12:36:34 UTC 2037]
Issuer: CN=GRuV
SerialNumber: [    5e484d26]]

Certificate Extensions: 1 [1]: ObjectId: 2.5.29.14 Criticality=false
```

圖9. 應用程式訊息頁面

4.2.4 程式碼安全

程式碼掃描一直是靜態分析中十分常用的測試方式，本研究採用的 PMD 即是程式碼分析工具，並透過自訂的 Ruleset，對 OWASP 的幾個弱點進行偵測。另一方面，APK 檔案反編譯後極有可能產生大量的待測檔案，造成整體測試時間的拉長，本研究除了透過待測 Queue 來對測試檔案進行排程工作之外，亦分析反編譯後的檔案架構，加速整體分析效能。圖 10 為程式碼安全掃描結果報表畫面，而圖 11 為偵測結果，可以看到此檔案包含不安全的加密演算法，即 OWASP 所定義的 A7 漏洞。

#	File	Method	Line	Issue	Problem
1	ListViewAutoScrollHelper.java	scrollTargetBy	61	< A1 - Injection >	localView.getTop of type UNKNOWN_TY information: Check whether UNKNOWN_
2	et.java	y	90	< A7 - Insecure Cryptographic Storage >	Insecure Cryptographic Algorithm
3	h.java	a	167	< A7 - Insecure Cryptographic Storage >	Insecure Cryptographic Algorithm

圖10. 程式碼安全檢測報表

```
88      try
89      {
90        Object localObject = MessageDigest.getInstance("MD5");
91        ((MessageDigest)localObject).update(paramString.getBytes());
92        localObject = String.format(Locale.US, "%032X", new Object[] {
93        return localObject;
94      }
95      catch (NoSuchAlgorithmException localNoSuchAlgorithmException)
96      {
```

圖11. OWASP A7 漏洞範例程式結果

五、 結論

本研究實作一個靜態分析平台，滿足經濟部工業局資訊安全檢測基準中的發布安全、使用者的敏感資料、程式碼安全等幾個項目，改善現有分析工具僅對部分問題做分析的弱勢，並且，本研究也進一步的將所使用的工具做功能優化，提供更佳的分析能力，以得到更全面、更可信的分析結果。

未來展望方面，在資訊安全檢測基準中，本研究僅對於做檢測，仍有許多延伸空間可以進行研究開發。另外，靜態分析的錯誤率仍有很大的改進空間，可以加上動態分析結果，或是使用多個分析工具的分析結果進行交叉比對，將能增加掃描正確率。

致謝

本研究由科技部計畫 104-2221-E-027-116 所補助，特此感謝。

參考文獻

[1] W. Enck, M. Ongtang, P. McDaniel, “On lightweight mobile phone application certification”, Proceedings of the 16th ACM conference on Computer and communications security, New York, NY, USA, pp. 235-245, 2009.

[2] Adrienne P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, “Android permissions demystified”, Proceedings of the 18th ACM Conference on Computer and Communications Security, New York, NY, USA, pp. 627-638, 2011.

[3] L. Davi, A. Dmitrienko, AR. Sadeghi, M. Winandy, “Privilege escalation attacks on Android”, Proceedings of the 13th International Conference on Information Security, Boca Raton, Florida, USA, pp. 346-360, 2011.

[4] K. W. Yee Au, Y. F. Zhou, Z. Huang, D. Lie, “ PScout: Analyzing the Android Permission Specification”, Proceedings of the 19th ACM Conference on Computer and Communications Security, New York, NY, USA, pp. 217-228, 2012.

[5] Androguard, https://github.com/androguard/androguard, Accessed September 14 2015.

[6] Androwarn, https://github.com/maaaaz/androwarn/, Accessed May 2 2016.

[7] AndroBugs Framework, https://github.com/AndroBugs/AndroBugs_Framework, Accessed May 2 2016.

[8] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Octeau, P. McDaniel, “FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps”, Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, New York, NY, USA, pp. 259-269, 2014.
[9] P. Faruki, V. Laxmi, A. Bharmal, M.S. Gaur, V. Ganmoor, “AndroSimilar: Robust signature for detecting variants of Android malware”, Journal of Information Security and Applications, Vol. 22, pp. 66-80, 2015.
[10] DroidAnalyst, http://droidanalyst.org/, Accessed April 22 2016.
[11] The Open Web Application Security Project, https://www.owasp.org/index.php/Main_Page, Accessed May 21 2016.
[12] The Open Web Application Security Project Top Ten Project, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, Accessed May 21 2016.
[13] The Open Web Application Security Project Mobile Security Project, https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks, Accessed May 19 2016.
[14] 行動應用 App 基本資安規範, http://www.communications.org.tw/news/policy/item/download/88_32891bb0faa8cab74e5afab83032bf37.html, Accessed February 2 2016.
[15] NIST Special Publication 800-163 Vetting the Security of Mobile Applications, http://dx.doi.org/10.6028/NIST.SP.800-163, Accessed February 2 2016.
[16] 經濟部工業局「行動應用 App 基本資安檢測基準」, http://www.communications.org.tw/news/policy/item/download/152_f1ff259b7be8a0f848e47a9a1ac3c1ca.html, Accessed April 26 2016.
[17] AndroidUnusedPermissions, https://github.com/MindMac/AndroidUnusedPermissions, Accessed May 3, 2016.
[18] Mobile-Security-Framework, https://github.com/ajinabraham/Mobile-Security-Framework-MobSF, Accessed April 19 2016.
[19] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, M. Smith, “Why eve and mallory love Android: An analysis of Android SSL (in) security”, Proceedings of the 19th ACM Conference on Computer and Communications Security, New York, NY, USA, pp. 50-61, 2012.
[20] MalloDroid, https://github.com/sfahl/mallodroid, Accessed May 3 2016.
[21] PMD, http://pmd.github.io/, May 16 2016.
[22] GDS PMD Secure Coding Ruleset, https://github.com/GDSSecurity/GDS-PMD-Security-Rules, Accessed May 16 2016.
[23] Android Developers - Manifest.permission, http://developer.android.com/intl/zh-tw/reference/android/Manifest.permission.html, Accessed March 25 2016.