國立交通大學

資訊科學與工程研究所

碩 士 論 文

Android 程式權限分析

Android Application Permission Analyze

研 究 生：陶嘉仁

指導教授：曾文貴　教授

中 華 民 國 一 百 零 一 年 八 月

Android 程式權限分析
Android Application Permission Analyze

研 究 生：陶嘉仁　　Student：Chia-Jen Tao

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

August 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年八月

Android 程式權限分析

學生：陶嘉仁　　指導教授：曾文貴

國立交通大學資訊科學與工程研究所碩士班

摘　　要

由於智慧型手機的普及，使得手機與生活中的關係越來越密切，許多人會使用手機上網、瀏覽電子郵件、購買商品等等關係到個人隱私資料的活動，將帳號密碼、信用卡號碼等個人資訊也儲存在手機上。Android 是一個開放原始碼的作業系統，因為它開放的特性，許多手機商都採用這個系統，也因此 Android 作業系統的普及性逐年增高。Android 的用戶可以輕鬆地從內建程式 Google Play 下載到各式各樣的 app 來使用，但是 Google Play 在 app 的安全檢查機制並沒有相當完善，所以使用者很容易就會暴露在資訊安全的危機當中。如果使用者沒有細心注意 app 的所有內容，很可能就會被存取到個人隱私資料，洩漏給惡意的第三方。我們實作出一個系統，從蒐集的資料建立資料庫，並且提供一個可以方便使用的 app 來幫助使用者檢測 app 所用到的手機資源，提供給使用者建議，幫助使用者判斷 app 的安全性。

Android Application Permission Analyze

Student：Chia-Jen Tao Advisors：Dr. Wen-Guey Tzeng

Department of Computer and Information Science
National Chiao Tung University

ABSTRACT

Since mobile phone plays an important role in life, more and more people surf the web pages, read e-mail, and shop on the internet by mobile phone. Users also store their personal information like address, account password, and credit card number in the phone. Android is an open source mobile operating system and many mobile phone manufacturers use it. Android users can easily download applications by a build-in software—Google Play. But Google Play has no the strict examination mechanism for the applications. For this reason, hackers can spread malicious applications to steal the personal information stored in the mobile phone. The users may download the malicious application and leak out the personal information unconsciously. We collect the applications, build a system to make database and write a examine program. The users can use the program based on the database to examine the application and get a clear and useful risk report to help them decide whether or not to install it.

誌　　謝

能完成這篇論文，最重要的是要感謝曾文貴教授，從碩一教導的密碼理論，一步步帶領我走進資訊安全和密碼理論裡面，讓我了解資訊安全的理論、實務和重要性。碩二起每周都從繁忙的行程中播出一到兩個時段幫助我研究，從論文方向的尋找，一直到研究過程中的指點和討論都對我有非常大的幫助。每周兩次的報告，讓我從生澀的表達到可以講清楚說明白的傳達自己的意見，無論是在往後工作或是生活上都有非常大的影響。

再來要感謝的是林孝盈學姊，沈宣佐學長和陳毅睿學長。從碩一起一起執行計畫，其中包括程式撰寫、修正和架設系統，沈宣佐學長都不吝找出時間和我一起研究討論，這篇論文合作的中華電信計畫，包括整個防禦情境的架構討論以及實作上的建議，陳毅睿學長都給我非常大的幫忙，論文的指導和報告的技術細節，林孝盈學姊都一一的指導和修正，讓我在口試報告的時候能夠順暢無礙。

最後要感謝我的同學們，賴託登、林輝讓、陳彥宇，和我在這兩年間不僅僅是同學關係，我們還是工作夥伴和室友關係，在生活中和學業中都給我莫大的幫助，非常感謝他們。

目　　錄

圖 目 錄

表目錄

第一章、介紹

1.1 智慧型手機介紹

智慧型手機能夠媲美一台個人電腦，它具有獨立的作業系統以及完善的使用者介面，它擁有很強的應用擴展性、能方便隨意地安裝和刪除應用程式，擁有高畫質觸控螢幕，能隨時使用觸控的方始進行輸入，能進行多項程式同時運作，並且擁有強大的多媒體、郵件、上網功能，能完全替代像 MP3、MP4 和 PDA 這樣的傳統多媒體設備。智慧型手機能替代個人電腦處理辦公事務和其他事務，它能與網路隨時保持連接，能隨時進入網路，並且能與電腦、筆記本電腦等其他設備同步資料。

智慧型手機所使用的作業系統包括 iOS、Android、bada OS、MeeGo、Palm OS、WebOS、Windows Mobile、Symbian OS 及 BlackBerry OS 等等。其中又 Android 為主流，Nielsen 報告指出，在 2010 Q2 時，Android 作業系統使用率占整體使用率一半以上[1]。

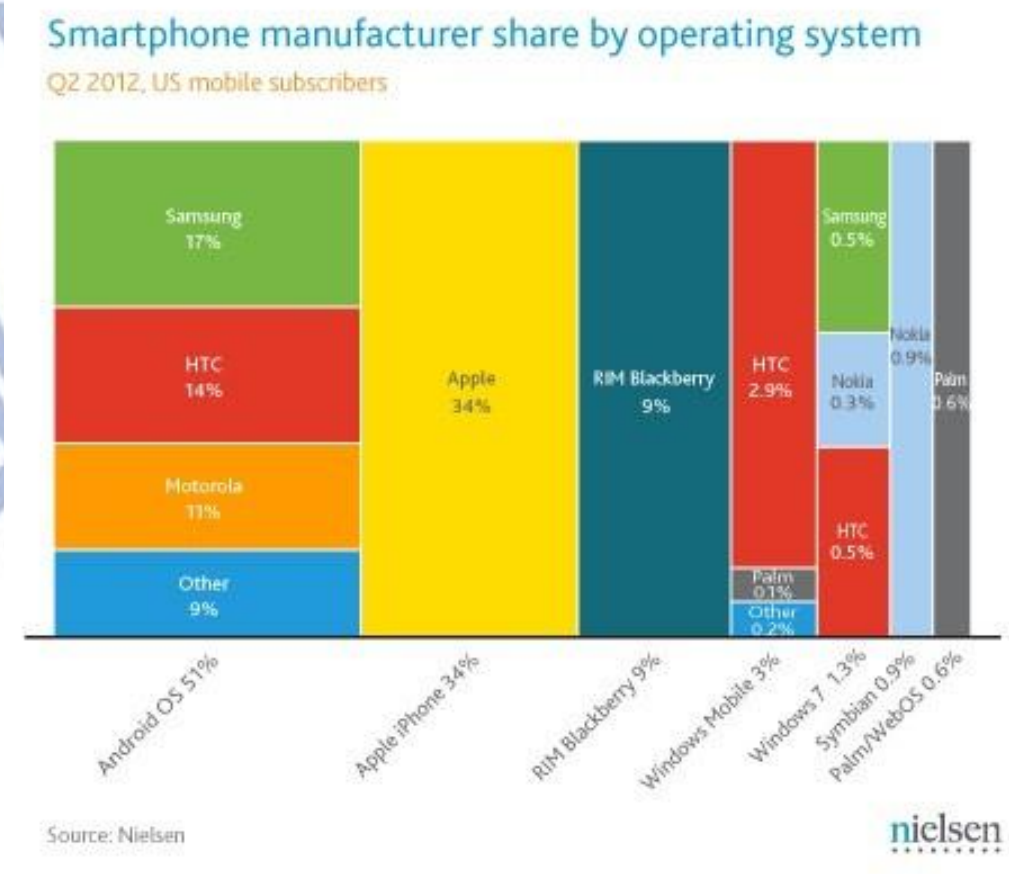


圖 1 智慧型手機使用率 (來源: Nielsen [1])

Android 是 Google 公司一個基於 Linux 核心的軟體平台和作業系統。Android 的特點是開放原始碼，它的 SDK 是開放給任何開發商，開發商都可以隨意更改介面。例如 HTC 的 HTC Sense、Samsung 的 TouchWiz 等等。2008 年開始，Google 就不斷更新 Android 的版本，分別推出 1.5Cupcake、1.6Donut、2.0~2.1Eclair、2.2Froyo、2.3Gingerbread、3.0 Honeycomb 及 4.0 Icecream Sandwich 等。

Android 作業系統使用了沙箱(sandbox)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(permission)送出給系統，並且將其所需 permission 以列表的形式展現出來，供用戶檢視。例如一個第三方瀏覽器需要「連接網路」的權限，或者一些軟體需要撥打電話，發送簡訊等 permission。用戶可以根據 permission 列表來考慮自己是否需要安裝，用戶只有在同意了應用程式 permission 之後，才能進行安裝。

圖 2 Permission 列表

2009 年 2 月，Google 推出 Android Market 線上應用程式商店(2012 年 3 月 6 日更名為 Google Play)，使用者可在該平台網頁尋找、購買、下載及評分使用智慧型手機應用程式及其他內容。第三方軟體開發商和自由開發者則可以通過 Android Market 發布其開發的應用程式。根據數據統計機構 Distimo 統計[2]，2012 年，Android Market 的活躍軟體數目已超過 40 萬，在去年的 4 月和 8 月，才突破了 20 和 30 萬大關，足見 Android Market 的軟件數目正急速增長。

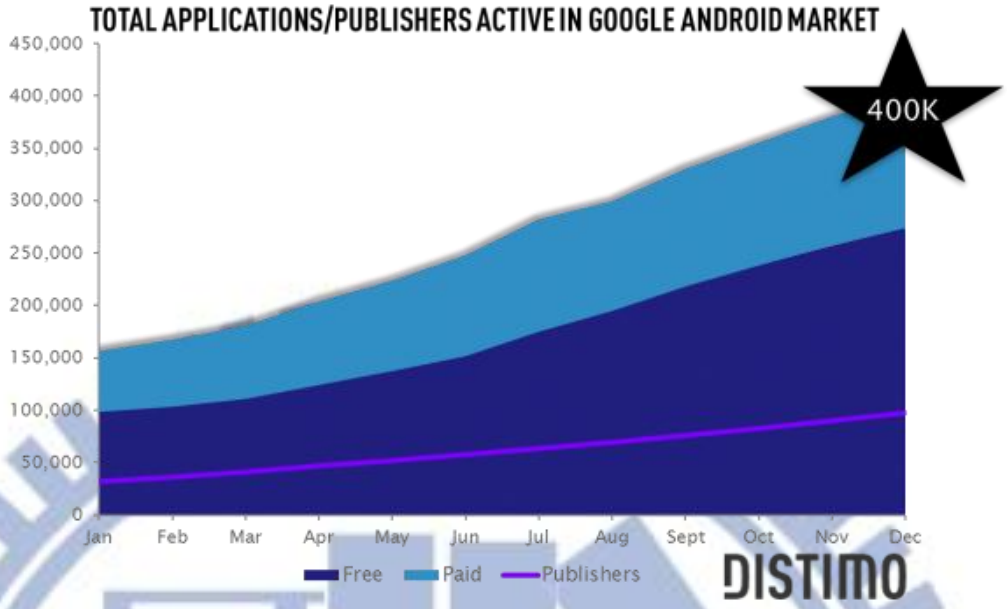


圖 3 Android app 數量 (來源: Distimo [2])

由於 Android 作業系統的開放和自由性，一些惡意程式和病毒也隨之出現。2010 年 8 月，卡巴斯基病毒實驗室報告稱發現了 Android 作業系統上首個木馬程式，並將其命名為「Trojan-SMS.AndroidOS.FakePlayer.a」，這是一個通過簡訊方式感染智慧型手機的木馬，並且已經感染了一定數量的 Android 設備。除了簡訊感染方式，這些 Android 木馬還可以偽裝成一些主流的應用程式，並且還可以隱藏在一些正規的應用程式之中。2011 年一月有一種名為 Geinimi 的木馬程式即為這類型的惡意程式。防毒軟體公司 Lookout Mobile Security 和賽門鐵克均表示，當用戶下載包含病毒的 app 時，手機將會被感染。Lookout 表示，一些可能含有病毒的軟件包括遊戲《Monkey Jump 2》、《President vs. Aliens》、《City Defense》和《Baseball Superstars 2010》。

當智慧型手機的用戶下載應用軟件或程式時，並不知道那些軟件和程式已經隱藏了這種病毒，一安裝便會「中毒」。被感染的手機將會每 5 分鐘連接一次遠端伺服器，隨後將會把手機位置、硬體 ID 和 SIM 卡信息等發給伺服器。

1.2 研究動機

我們希望做出一個可以幫助使用者判斷 permission 合理性的一個程式，和其他研究不同的是我們希望不要更動系統，也不需要使用者提供大量的資料(例如下載程式的網頁內容，app 檔案)，就可以幫助使用者判斷 app 是否有用到不該使用的 permission，而不用將所有 permission 的說明細項一一閱讀。由於目前 android 還尚未讓使用者個別開放或是關閉 permission，所以在提供結果報告後，使用者可以清楚的了解安裝此 app 會有甚麼風險，如果能接受所有的話，再安裝此 app。

1.3 相關研究

近年來智慧型手機大量出現，手機內資料的安全性也越來越受到重視，各種相關的研究也一一出現。分析手機程式主要分為兩個方向：動態分析和靜態分析。

在動態分析中，Isohara et al. (2011) [3]發表了一篇由系統 kernel 行為來監視 app 是否為惡意程式的一篇論文，藉由監測 app 以及 kernel 所執行的 function call，來了解 app 存取了那些隱私資料。首先在系統中記錄所有執行的 function call，再藉由預先定義好的 signature 來做比對，分析出結果給使用者。

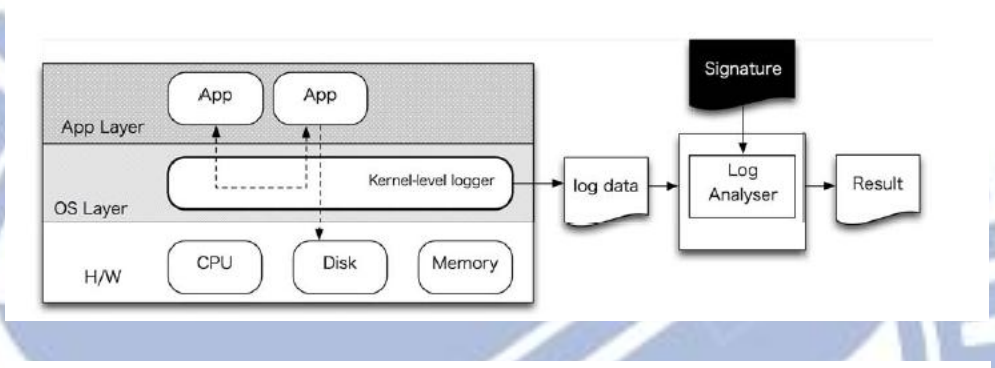


圖 4 Isohara 系統架構圖

Enck et al. (2010) [4] 發表了一篇藉由標記隱私資料來保護手機內容的論文。利用 tainting 的技術，將隱私資料標註起來，並且追蹤(tracking)資料流向，當需要使用的時候，再藉由預先定義好的規則(Policy)，來判斷是否要給 app 使用這個資料。和原本的 file system 比較起來大約會多消耗 26%的時間，而且使用者必須修改系統來應用這個機制。

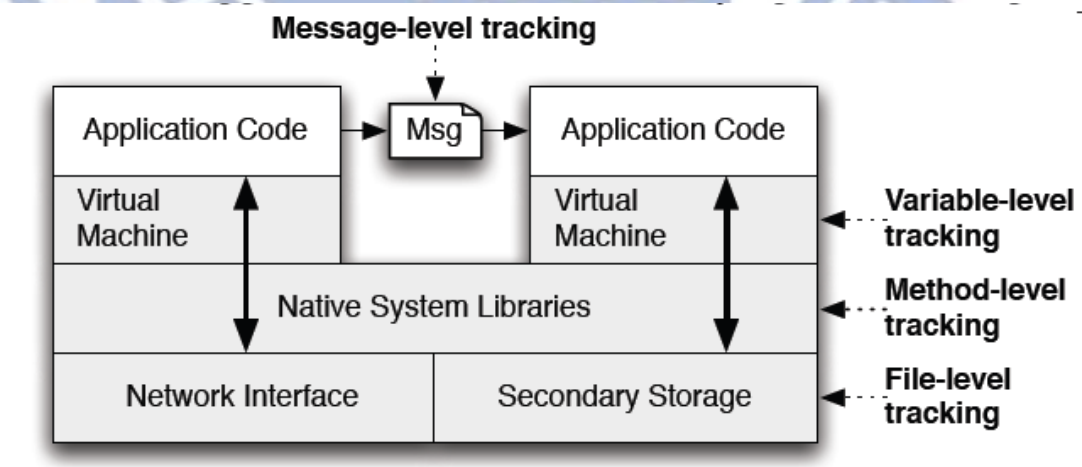


圖 5 Enck OSDI 2010 系統架構圖

Shahzad et al. (2011) [5]發表了一篇藉由分析 app 在執行期間所占用的資

源來判斷是否為惡意程式，他主要分析下列幾個系統資源：

1. as_users

   Number of processes using current address space

2. page_table_lock

   Used to manage the page table entries

3. hiwater_rss

   Number of page frames a process owns

4. shared_vm

   Number of pages in shared file memory mapping of a process

5. exec_vm

   Number of pages in exec. memory mapping of process

6. nr_ptes

   Number of page tables owned by a process

藉由 linux 的惡意程式資料庫，發現可以從以上幾個資料，找出惡意的程式，希望藉由這項結果可以運用在 andorid 的平台上。

在靜態分析方面，Enck et al. (2011) [6]發表藉由將 app 解壓縮，再分析其中的原始碼來找出 app 要利用到的隱私資料，他們蒐集了 1100 個 app 的程式，藉由分析其中的原始碼，發現了有 27 個程式會將個人隱私資訊洩漏給廣告公司或是惡意攻擊的伺服器主機。

Barrera et al. (2010) [7] 發表了一篇研究 permission 分布的一篇 paper，他在其中利用了 SOM (Self-Organization Map) 來展示各種 permission 的分布不規則性，大多數的 permission 只會出現在特定的幾個類別(type) 中，此一發現也是我們接下來想要實作出程式幫助使用者的其中一個原因。

另外還有其他論文希望可以幫助使用者拒絕提供某些 permission，像是 Enck et al. (2009) [8]有提到可以藉由修改安裝檔來幫助使用者拒絕某些 permission，但是修改後如果要再恢復就必須要重新安裝，而且修改過後的 app 無法在 Android Play 上自動更新。

Nauman et al. (2010) [9] 提出了藉由修改系統，可以動態的讓使用者選擇是否要開放 permission，讓使用者可以在程式安裝時，設定開放或是有條件開放 permission 給 app 使用。

圖 6 Nauman 系統架構圖

Batyuk et al. (2011) [10] 發表的論文中，設計了一個網站，可以讓使用者上傳 app，並且藉由 decompile 從中取得所使用的 API call，以及將所利用到的廣告贊助商，和 permission 列表一一讀取出來。

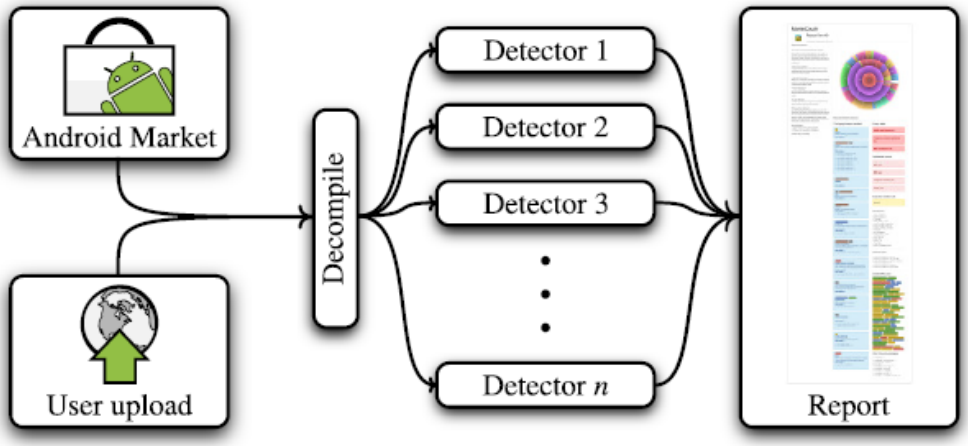


圖 7 Batyuk 系統架構圖

1.4 成果重點

以往的研究多專注在分析程式碼，或是修改系統來增加安全性，使用者需要專業的 Android 知識或技術，包括需要去解壓縮 app 的檔案，或是修改重灌整個 Android 系統，對使用者來說有很大的入門門檻。而我們設計出的程式，是可以不用取得 app 的原始檔案以修改系統，就可以輕鬆地使用而不需要 Android 的專業技術。我們藉由蒐集 app 來建立資料庫，利用資料分析的方法提供給使用者三種面向的分析報告。使用者可以從我們的分析報告中得到有關 app 使用的手機資訊和資源，進而判斷是否要安裝 app。

1.5 各章重點介紹

第一章介紹 Android 的發展，以及相關資訊安全的研究概況，然後提出我們的想法，實作出的成果以及和其他研究發展的不同之處。第二章對於 Android 的安全防護系統 Permission 做一個簡單的介紹，然後介紹架構我們系統建立資料庫所需要的工具，以及我們整個系統的一個介紹。第三章介紹我們的程式是如何提供使用者結果報告，其背後的原理、想法和做法。第四章介紹我們系統的實作步驟以及程式分析的流程。第五章是我們實做出來的程式搭配系統產生的資料庫提供給使用者報告的實際例子和說明。第六章做一個總結以及未來希望能再更進步的一些想法。

第二章、系統介紹

2.1 Android Permission 介紹

Android 系統下，為了避免錯誤或是惡意的濫用，app 能取得的手機資源是有限制的，有一部分的功能是被限制，只有 system 可以使用(例如 SIM 卡的操作)，沒有提供任何的 API 給外部的 app 使用，另外一部分是只能由電信公司，或是手機商預先安裝好的程式才能使用，最後才是可以供一般 app 使用的功能，而這些功能是由一個 permission 機制所保護。

Permission 保護的 API：

- Camera functions
- Location data (GPS)
- Bluetooth functions
- Telephony functions
- SMS/MMS functions
- Network/data connections

這些功能被限制只能經由 OS 的 API 才能利用，而要利用這些功能的程式必須要有相對應的 permission。在程式安裝前，必須要提示使用者所有 app 可能會利用到的 permission，由使用者去決定是否要安裝，如果答應安裝，則視同答應給予這些 permission。

Android 內定的 permission 很多，至 API level 14 為止共有 122 個，其中主要分成三個部分：

1. 關於個人資料保護(ex. contacts，account)，若是任意授權給 app 使用，可能會造成隱私洩漏問題。
2. 關於系統設定(ex. wifi，bluetooth)，app 取得這些權限可能可以竄改系統設定造成當機或是消耗手機電量，減少使用時間。
3. 關於手機功能(ex. camera，call phone)，這些權限提供給 app 利用手機上的功能，但惡意程式可能可以用來蒐集個人資料(攝影或是錄音)或是讓使用者產生額外的費用(發簡訊，打電話)。

當使用者要下載安裝 app 的時候，Google Play 會提示使用者所需的 permission，但是使用者並不一定很熟悉系統設定或是專業詞彙，無法容易的判斷是否要給予 permission，內建的 permission 數量繁多且功能繁雜，因此我們希望能提出一些建議幫助使用者判斷，讓使用者可以清楚的了解安裝此 app 會有甚麼風險，再決定是否安裝 app。

2.2 AndroidManifest.xml 文件結構介紹

在 Android 執行一個 app 之前，必須先知道該 app 有哪些元件 (Activity、Service 等)，這些元件的描述被放在 AndroidManifest.xml 檔案中。AndroidManifest.xml 會連同執行碼與相關資源等一同打包到附檔名為 apk 的專案檔中。(往後提到 apk 是指檔案本身，app 則為應用程式。)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <uses-permission />
    <permission />
    <application android:permission="...">
        <activity android:permission="..."
                  android:exported="...">
            <intent-filter>
                ...
            </intent-filter>
        </activity>
        <service android:permission="..."
                 android:exported="...">
            <intent-filter>
                ...
            </intent-filter>
        </service>
        <receiver android:permission="..."
                  android:exported="...">
            <intent-filter>
                ...
            </intent-filter>
        </receiver>
        <provider android:permission="..."
                  android:readPermission="..."
                  android:writePermission="..."
                  android:exported="...">
            <grant-uri-permission />
        </provider>
    </application>
</manifest>
```

圖 8 AndroidManifest.xml 基本架構

上面是一個 AndroidManifest.xml 的基本架構，下面對每一個部份做介紹：

- uses-permission

  請求你的 app 運作所需賦予 permission。這個部分就是我們需要的 permission 資訊。

- permission

  定義使用這個 app 提供的資源所需要的 permission。

- application

  一個 AndroidManifest.xml 中必須含有一個 Application 標籤，這個標籤聲

明了每一個應用程序的組件及其屬性(如 icon, label, permission 等)

- activity

  activity 是用來與使用者互動的主要工具。當用戶打開一個應用程序的起始頁面時就是一個 activity，大部分被使用到的其他頁面也由不同的 activity 所實現並聲明在另外的 activity 標記中。

  另外，爲了支援運行時系統查找你的 activity，你能包含一個或多個<intent-filter>元素來描述你 activity 所支援的操作。

- receiver

  intentReceive 能使得 app 收到數據的改變或者發生的操作，然後做出反應，即使它當前不在運行。利用 activity 標記，你能選擇地包含一個或多個 receiver 所支持的<intent-filter>元素。

- service

  service 是能在背景運行任意時間的組件。利用 activity 標記，你能選擇性地包含一個或多個 receiver 所支持的<intent-filter>元素。

- provider

  ContentProvider 是用來管理數據並發佈給其他應用程序使用的元件。

2.3 使用工具介紹

2.3.1 Android SDK & ADT

Android SDK 將開發 Android 程式所需要的所有元件都包含在裡面，裡面包含有：

SDK Tools：包含了開發程式需要的 Debug 工具，測試工具，是最基本必備的元件。

SDK Platform-tools：含有跟電腦連接測試的工具 adb，以及讀取 app 資訊的元件 aapt，這些在後面我們將會用到。

SDK Platform：裡面有 Android 所提供的 API。

System Images：提供了各種版本的原生 system OS 映像檔(Image)可供我們使用

還有其他 document、Samples for SDK、Sources for Android SDK 等等資源，是開發程式過程中需要的資料。此外我們還需要 Android Development Tools (ADT)，這是一個 Eclipse 的 plugin，裡面包含介面化的程式設計空間，下面的 DDMS 就是 ADT 提供的 tools 其中之一。

2.3.2 DDMS

DDMS 的全名是 Dalvik Debug Monitor Service，它為我們提供例如：為測試設備擷取畫面，針對特定的進程查看正在運行的程式以及堆(stack)信息、Logcat、廣播狀態信息、模擬電話接聽以及接收 SMS、虛擬地理坐標等等。DDMS 對 Emulator 和外接測試機有同等效用。如果系統檢測到它們(VM)同時運行，那麼 DDMS 將會默認指向 Emulator。

DDMS 有整合在 Eclipse 的 ADT 中，也有包含在 Android 的 SDK 中，所以有兩個方法可以啟動它：

From Eclipse： Window > Open Perspective > Other... > DDMS.

From the command line：

Type ddms (or ./ddms on Mac/Linux) from the tools/ directory.

DDMS 有提供一個 File Explorer，可以讓我們對手機或是 Emulator 中的檔案進行操作，我們將利用這個功能從手機中取得從 Google Play 下載下來的 app。

2.3.3 apktool

APK 是 Android Package 的縮寫，即 Android application package 文件或 Android 安裝包。每個要安裝到 Android 平台的應用程式都要被編譯打包為一個單獨的文件，副檔名為 apk。APK 文件是用專業軟體 eclipse 編譯生成的文件包，其中包含了應用的二進制代碼、資源、配置文件等。將 APK 文件直接傳到 Android 手機中執行即可安裝。

即使使用了解壓縮軟體取得上面這些資料，依然是無法進行閱讀的，為了程式執行的速度，所以這些資料都已經被編碼，我們要取得資訊就必須要利用軟體進行 decode，這就是我們要用 apktool 的目的了。

2.3.4 Weka

Weka 是基於 java，用於數據挖掘和知識分析一個平台。來自世界各地的 java 愛好者們都可以把自己的算法放在這個平台上，然後從大量數據中發掘其背後隱藏的種種關係。

我們用這個軟體，使用 data mining 的方法分析資料，裡面已經有實作出 SVM 的模組，我們只需要將資料整理成 Attribute-Relation File Format (ARFF)格式，就可以幫助我們算出要的結果。

2.4 實作系統介紹

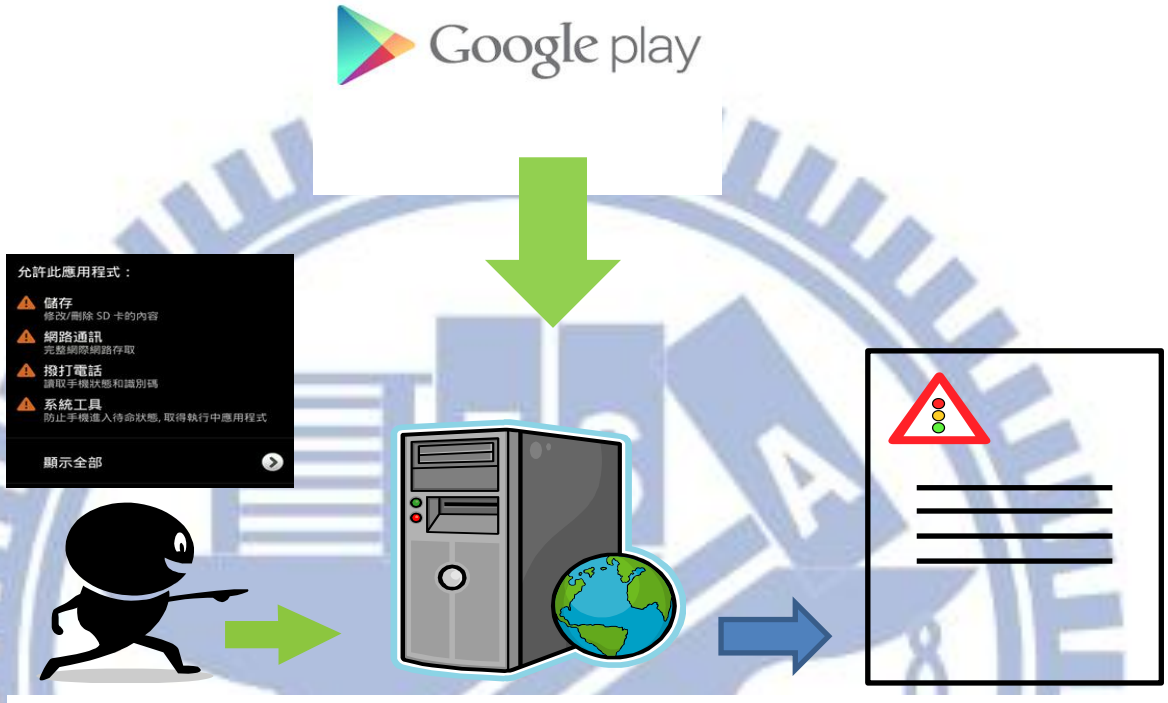


圖 9 實作系統架構圖

首先，我們從 Google Play 中下載 app，我們挑選熱門下載且下載次數較多，評價較高的 app 來建立我們的資料庫，一方面可以確保下載軟體是經由大量使用者認可的安全軟體外，另一方面也能確保分類的正確。下載後的 app 經過解壓縮掃毒後，讀取所需的 permission 建立資料庫，提供我們建議使用者的依據。當使用者想要安裝一個新的 app 時，可以將 app 所要求的 permission 列表以及 app 的類別(type)提供給我們，經由我們建立的資料庫，提供給使用者有用的訊息，幫助使用者輕鬆判斷是否要安裝 app，而不需要將軟體上傳，或是具備 Android 系統設定或是專業詞彙的知識。

第三章、方法介紹

3.1 系統介紹

我們系統將依照下面三個部分，以三個不同的面向，提供使用者建議以及判斷參考：

1. 將 Android 內定的 permission 依照功能分類，當程式使用 permission 時，可以直接提示使用者開放的個人資料、功能，或是可能更動到的系統設定，讓使用者可以清楚地了解風險而不用去細讀每個 permission 代表的意義。
2. 從 Google Play 收集各類程式(依照 Google Play 上定義的 type)建立資料庫。從資料庫中可以統計出各個 permission 使用的次數和比率，在使用者安裝 app 的時候，若是有出現少用的 permission 就要優先注意是否合適。
3. 利用資料分析的方法建立 permission 和 type 之間的關係。因為同一種 type 中的程式功能應該大致類似，所需要用到的手機資源也互相類似，由此來協助使用者判斷程式所使用的 permission 是否合理。

3.2 實作方法

3.2.1 permission 分類

Android 內定的 permission 主要是保護系統提供的資源，主要可以分成三大部分：

i. 個人資料：有關個人隱私資料有關的 permission。
ii. 系統設定：有關系統調教設定的 permission。
iii. 系統功能：手機可供 app 利用的功能。

此外我們整理出一些可能會造成系統當機或是嚴重危害的 permission，這類 permission 不應該出現在 app 中。

3.2.2 permission 使用頻率以及危險指數計算

前一節中的分類只是將所有的 permission 影響的部份整理出來，我們希望利用一些統計的數據幫助使用者判斷 permission 是否合適。由於 permission 代表的是手機的功能和使用到的個人資訊，我們希望能蒐集資料，比較出來功能較為特殊或是利用較多個人資訊的 app，在使用者安裝這些 app 的時候提出建議，提醒使用者再次檢查 app 的來源以及使用的資源。

我們蒐集 app，建立各個 permission 出現的頻率，使用者輸入 app 所需的

permission 時，我們可以由蒐集的 app 中統計出現率較低的 permission 警告使用者注意。

此外，我們想計算出一個危險指數(risk index)代表這個 app 危險的程度，危險指數越高代表 app 和其他 app 比較起來，使用較多的手機資源或是取得較多的個人資訊。常出現的 permission 就給他比較低的權重(weight)，代表這個 permission 是 normal 的，比較少出現的 permission 則有比較高的權重，代表其功能較為特殊，不是一般程式會利用到的。

因為以上理由，所以我們給每個 permission 一個權重，和他的出現頻率相反，出現率越高權重越低，然後我們利用距離公式計算 app 的危險指數，代表 app 和完全沒使用 permission 的 app 間的距離，定以下危險指數的公式：

Permission 的權重：$(1-R(p))^2$，$R(p)$：permission p 的出現頻率

Index(P)=$\sqrt{\sum_{p\in P}(1-R(p))^2}$，P={$p_1$,$p_2$,…}：app 使用的 permission

由此指數來衡量此 app 的危險程度，若是沒有使用 permission 的 app 則指數為 0。

由於使用者在安裝程式的時候應該知道程式的功能和用途，因此我們希望藉由使用者提供我們程式功能的類別(type)，更詳細的分析 permission 是否合理。

我們將前面的危險指數計算方法做一些調整，將出現頻率調整成計算在同一類別裡的出現頻率：

$R(p,t)$：permission p 在 type t 的出現頻率

Index (P , t)=$\sqrt{\sum_{p\in P}(1-R(p,t))^2}$

用這樣計算出來的指數，可以跟資料庫同類別 app 之間做比較，針對不同的類別我們可以設定不同的標準，來衡量使用者 app 的危險程度。

3.2.3 permission 與類別

利用 data mining 的方法，建立 type 間的判斷準則，協助使用者判斷。我們將各類程式兩兩一組，利用 data mining 中分類(Classification)演算法，由機器去找出判斷這兩類數據的準則，利用這個結果以及使用者提供的類型，我們可以協助建議是否有不合適的 permission。下面介紹使用的分類演算法概念以及如何利用結果。

Support Vector Machines(SVM)是一種分類演算法，由 Boser et al. (1992) [11]根據統計學習理論提出的一種機器學習方法。簡單來說，SVM 想要解決以下的問題：找出一個超平面(hyperplane)，使之將兩個不同的集合分開。為什麼使用超平面這個名詞，因為實際資料可能是屬於高維度的資料，而超平面意指在高維中的平面。

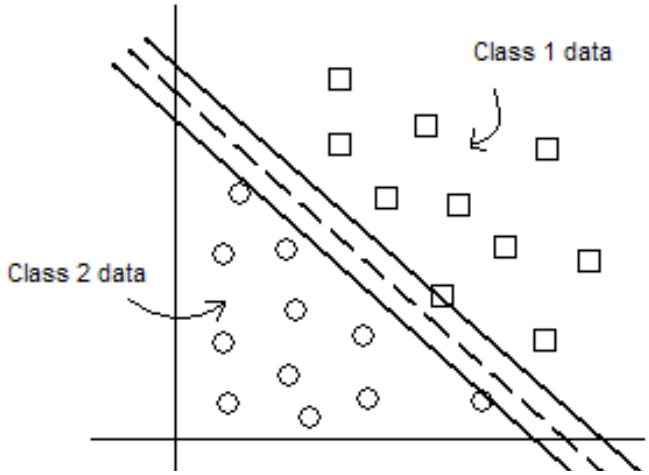


圖 10 SVM 數學示意圖

以二維的例子來說，如上圖，我們希望能找出一條線能夠將圓點和方點分開，而且我們還希望這條線距離這兩個集合的邊界(margin)越大越好(即圖中兩條黑色實線距離)，這樣我們才能夠很明確的分辨這個點是屬於那個集合。

接下來我們將用更數學一點的方式來描述上面的問題，假設我們有一堆點集合$\{x_i,y_i\}$，$i=1,\ldots,n$，$x_i=R^d$，$y_i=+1$、-1。點集合是我們所選出來的兩類 app's record，X_i 代表的是所宣告的 permission 情況(例如 $X_1=(1,0,0,1)$代表 app1 使用了第 1 跟的 4 個 permission，而現實情況這個向量會是多維的)，y_i 則代表著我們所選出來的兩類(一類的 app 全部設為 1，另一類全設為-1)。我們希望能找到一條直線$f(x) = w^T x - b$，使得所有 $y_i=1$ 的點落在 $f(x)>0$ 這一邊，$y_i=-1$ 則落在另一邊，這個直線中的 w 向量就可以幫助我們判斷 permission 應該是屬於哪一類。其中 $w_k>0$，則代表 permission k 較具有 $y_i=1$ 這一類的特性，$w_k<0$，則代表 permission k 較具有 $y_i=-1$ 這一類的特性。

因此利用此方法，我們將資料庫中每兩類一組做一次這個方法，當使用者輸入一個 app 以及類別 t 時，我們將類別 t 和其他各類所產生的$f(x)$來做分析，是否此 app 有用到具有其他類別 t'特性的 permission，可以警告使用者此 app 應該具有 t'的功能，否則可能是用了不適當的 permission。

3.3 分析資料和利用

使用者會提供我們一個 app 使用的 permission，以及這個 app 的 type。

1. 由 permission 分布提供建議

從資料庫中我們統計出各個 permission 在各個 type 中的使用率，利用這個分佈，我們提供給使用者建議。當使用者輸入使用的 permission 和 type，查找這個表每個 permission 對應的分布情況，設定一個標準，如果高於這個標準，表示這個 type 有足夠多的程式使用這個 permission，則判定為安全。如果低於標準，表示這個 type 不應該出現這個 permission，判定為危險，另外回報使用者危險的 permission 應該是出現在哪幾類。此外我們提供一個參考的危險指數來衡量這個程式的危險性，對比於同類型的平均值和標準差，來做出判斷。

下面舉一個簡單的例子：

	Type A	Type B	Type C
P1	0.1	0	1
P2	0.5	0.5	0
P3	0	1	1
P4	1	0	1

表 1 Distribution 範例

P1~P4 是四個 permission，P1 在有 10%的 type A app 使用，沒有 type B app 使用，以此類推。假如我們標準訂在 0.2，也就是要 20%以上的 app 使用才安全。

i. (1,0,0,1)，type A，輸出：P1 危險，應為 type C

ii. (0,0,0,1)，type B，輸出：P4 危險，應為 type A 或 type C

iii. (1,0,1,0)，type C，輸出：安全

下面是實際程式的例子：

Input: com.real.RealPlayer-1.apk Media

Danger: BLUETOOTH

Danger: BLUETOOTH_ADMIN

Danger: SYSTEM_ALERT_WINDOW

Danger: RECEIVE_BOOT_COMPLETED

Suggest type: Tool, Communication, Productivity

Danger: WRITE_SETTINGS

Suggest type: Music, Productivity, Communication

這個例子使用了危險的 BLUETOOTH ADMIN 權限，會修改系統設定以及系統警告視窗的 permission，我們提供建議是要有 Music，Productivity，Communication 這三類型功能的才會用到這個 permission。最後是此 app 的指數，高於平均 1.4 個標準差，要提醒使用者特別注意這個程式的正當性。

2. 由分類演算法提供建議

我們將資料庫的所有 type 兩兩一組，讓系統自動學習分辨的函數，當使用者輸入資料 **x**(permission 使用情況)和類別 t 後，將 **x** 代入 t 和其他各類的產生的函數做判斷，如果系統判定為 t 以外的類型，則可以利用這個結果來建議使用者。

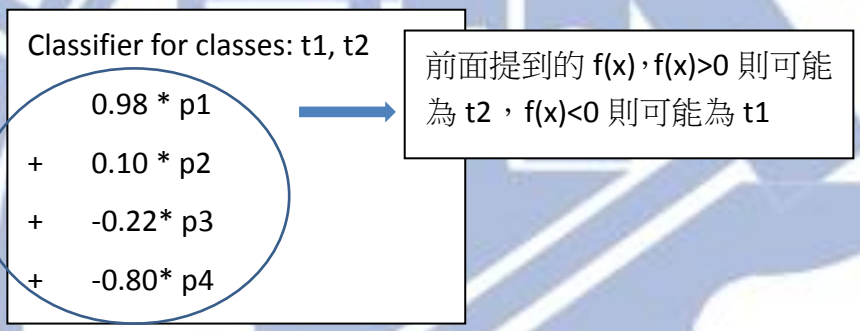


圖 11 Classfication 範例

下面舉一個簡單的例子：

使用者輸入這個 APP 使用了 permission1 和 permission2，則 x=(1,1,0,0)，以及 type 2。我們將 x 代入 f(x)=0.98>0，則我們可以認為這個 app 使用的 permission 情況適合這個類別。如果這個 app 是 type 1 則不適合，我們可以回報使用者建議的類別以及不適合的 permission。

這個分類法產生出來的結果，我們還可以用來比較 permission 在兩類之中較具有哪一類的特性，如果權值是<0 則較具有 t1 特性(如上例的 p4,p3)，權值是>0 則較具有 t2 特性(如上例的 p1,p2)

下面是真實結果的一部分：

```
6949  Classifier for classes: Personalization, Travel
6950
6951  BinarySMO
6952
6953  Machine linear: showing attribute weights, not support vectors.
6954
6955           0.6431 * ACCESS_COARSE_LOCATION
6956   +       1.1857 * ACCESS_FINE_LOCATION
6957   +       0      * ACCESS_LOCATION_EXTRA_COMMANDS
6958   +       0.2606 * ACCESS_MOCK_LOCATION
6959   +      -0.5346 * ACCESS_NETWORK_STATE
6960   +      -0.0763 * ACCESS_WIFI_STATE
6961   +       1.7642 * CALL_PHONE
6962   +       1.2521 * CAMERA
6963   +      -0.5538 * CHANGE_COMPONENT_ENABLED_STATE
6964   +       0      * CHANGE_CONFIGURATION
6965   +       0      * CHANGE_WIFI_STATE
6966   +       0.241  * DISABLE_KEYGUARD
6967   +      -0.9186 * GET_ACCOUNTS
6968   +      -1.6445 * GET_TASKS
6969   +       0.6133 * INSTALL_PACKAGES
6970   +       0.7064 * INTERNET
6971   +       0      * MODIFY_AUDIO_SETTINGS
6972   +       0.8035 * MOUNT_UNMOUNT_FILESYSTEMS
6973   +      -0.4677 * READ_CONTACTS
6974   +       0.0291 * READ_PHONE_STATE
6975   +      -1      * READ_SYNC_SETTINGS
6976   +       0.0211 * RECEIVE_BOOT_COMPLETED
6977   +       0.241  * SEND_SMS
6978   +       0.2311 * SET_ORIENTATION
6979   +      -0.2919 * SET_WALLPAPER
6980   +      -0.2919 * SET_WALLPAPER_HINTS
6981   +      -0.2012 * SYSTEM_ALERT_WINDOW
6982   +      -0.3833 * VIBRATE
6983   +      -0.3074 * WAKE_LOCK
6984   +      -0.7087 * WRITE_CONTACTS
6985   +       0.3822 * WRITE_EXTERNAL_STORAGE
6986   +       0.0655 * WRITE_SETTINGS
6987   -       0.9999
6988
6989  Number of kernel evaluations: 3279 (90.537% cached)
```

圖 12 Classification 實例

第四章、系統實作

4.1 蒐集資料

4.1.1 修改手機取得 root

因為 Google Play 不直接提供 app 的下載服務，一定要直接安裝在手機上，所以為了收集 app 的 apk 檔，我們要先安裝在手機上後再取出來。這些安裝在手機上的 apk 檔必須要手機有 root 權限才能取得，一般的 Android OS 是不提供這個權限，所以要先修改 OS，取得 root 權限後再由 DDMS 軟體取出來。

取得 root 權限有兩個方式，第一是利用網路上 root 的程式，它是利用系統的漏洞進而取得 root 的權限。第二是修改 OS，直接使用客製化的 OS 而非手機原生的系統，這樣我們可以修改任何系統中的功能和限制，讓我們研究更便利。我們這邊採用第二種方法。

使用手機： HTC Hero，4G SD Card

使用 OS： Elelinux-7.1.0

從 Google Play 下載 ROM Manager 這個軟體，藉由這個軟體，我們可以刷入客製化的 OS，在這裡我們使用 Elelinux-7.1.0 這個 OS。

從網路上下載好 Elelinux-7.1.0 後放入記憶卡中，再由 ROM Manager 這個軟體中選擇 Install ROM from SD Card，就會自動重新開機並且刷入客製化的 OS，藉由這個 OS 我們就可以讀取所有系統的資料。

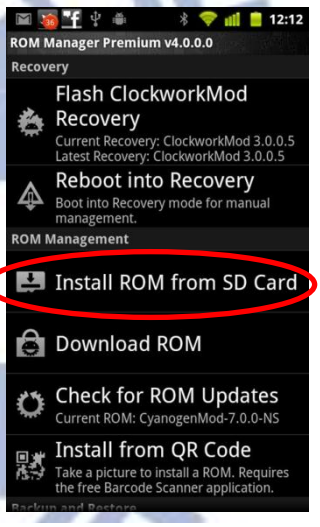


圖 13 Root 手機

由於新版的 toolbox 並沒有 linux 指令 ls，所以在使用 DDMS 時會生讀不到資料的問題，我們只要將原生的 toolbox 取代客製 OS 的就好。

1. 先開一個模擬器，套用 SDK 附的原生映像檔，然後將 /system/bin 中的 toolbox 抽出來(這個可以用 DDMS 操作，因為是模擬器，我們可以讀到裡面資料)
2. 將手機接上電腦，在命令提示字元中，切換到 Android SDK 下的 platform-tool 資料夾，執行 adb remount
3. adb push c：\toolbox /system/bin (c：\ 是你剛抽出來的 toolbox 所放置的路徑，請自行修改)
4. adb shell

5. cd /system/bin
6. chmod +x toolbox
7. ln -s toolbox ls
8. exit
9. 重新啟動裝置

4.1.2 從手機中取得 app 的 APK 檔案

從 Google Play 下載檔案，它會自動存放於兩個位置底下

1. /data/app 中 xxx-1.apk，xxx 是 app 的名字。
2. /mnt/asec 下 xxx 資料夾，但是我們要的是他的 apk 檔，所以我們取裡面的 pkg.apk 檔。
3. 還有一種是在/data/app 存成 xxx-1.zip，等同於 apk 檔，只是附檔名不同

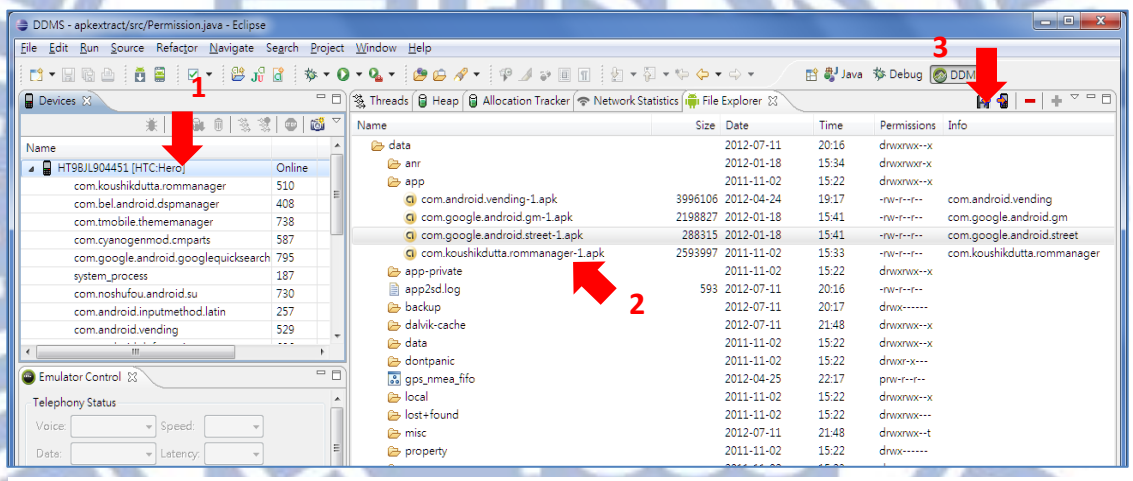


圖 14 DDMS 操作

然後經由 DDMS 介面就可以一次把需要的檔案全部 pull 回來。

1. 點選裝置
2. 選擇多個 apk 檔案
3. 按下 pull 儲存

我們將同一個類別的 apk 儲存在同一個資料夾下，每個資料夾就代表著一個類別，以利管理。

4.1.3 收集的資料

我們從 Google Play 上下載 18 個類型共 875 個熱門 app，分類則依照 Google Play 的分類為主。

Type	#	Type	#
Books	19	News	29
Business	46	Personalization	45
Communication	76	Photography	41
Education	29	Productivity	79
Entertainment	54	Shopping	37
Finance	52	Social	50
Game	47	Tool	42
Media	42	Transportation	73
Music	55	Travel	59

表 2 蒐集 app 數量

4.1.4 解譯和建立資料庫

收集了 app 後我們利用 apktool 將程式解譯，然後讀取 app 裡面的 AndroidManifest.xml 檔案，建立資料庫

apktool 的解譯指令：

apktool d xxxxxx.apk ABC

這裡「d」表示要解碼。xxxxx.apk 是要解壓縮的 APK 文件。ABC 是子目錄名。解壓縮的文件會放在這個子目錄內。

解壓縮後我們可以讀取 AndroidManifest.xml 這個文件，取得這個 app 所需的 permission。

我們寫了一個程式，會將同一資料夾下的所有 apk 檔利用 apktool decode，然後可以產生每個類別的文件，裡面記錄著同一類別下 permission 的使用情形。在文件中，第一列是總共 app 的個數，第二列起是每個 app 的 permission 使用情形，每一列的紀錄如下：(p_0,p_1,p_2,…,p_{121},p_{122},name)，p_0~p_{121} 為 android 內建 122 個 permission，有使用則計做 1，未使用為 0，p_{122} 記錄著自行定義的 permission 個數，所以為>=0 的整數，name 則為 string，記錄 app 名稱。

所有 type 的 app 記錄就構成了我們的資料庫。

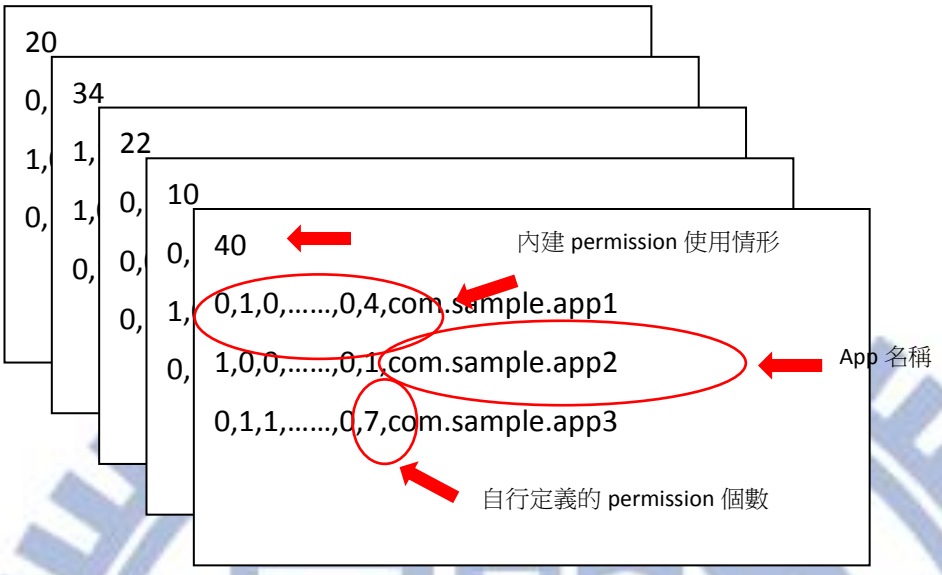


圖 15 資料庫示意圖

4.2 Weka 的使用方式

使用的版本是 weka 3.6.7。

```
% comment
%
@RELATION name


@ATTRIBUTE att1    NUMERIC
@ATTRIBUTE att2    NUMERIC
@ATTRIBUTE att3    {t1,t2,t3}


%The Data of the ARFF file looks like the following:


@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
```

圖 16 arff 檔案格式

主要分三個部分：

1. @RELATION 後面接 arff 的名字
2. @ATTRIBUTE 屬性名字以及類型，在這裡我們填入各個 permission 的名字，類型則用列舉法{0,1}，0 代表沒使用，1 代表有使用。最後要加入這支 app 的 type，供機器自動學習。
3. @DATA 下一行開始填入我們的資料，格式必須要一一對照上面 ATTRIBUTE 所設定的內容。

接下來我們要利用 Weka 裡面已經實作的 SVM 演算法—SMO 來幫我們算出各類別間的判斷式 f(x)。

1. 啟動 Weka，然後選擇 Explorer。

圖 17 啟動 Weka

2. 然後選擇上方的 Open 載入剛做好的 arff 檔案

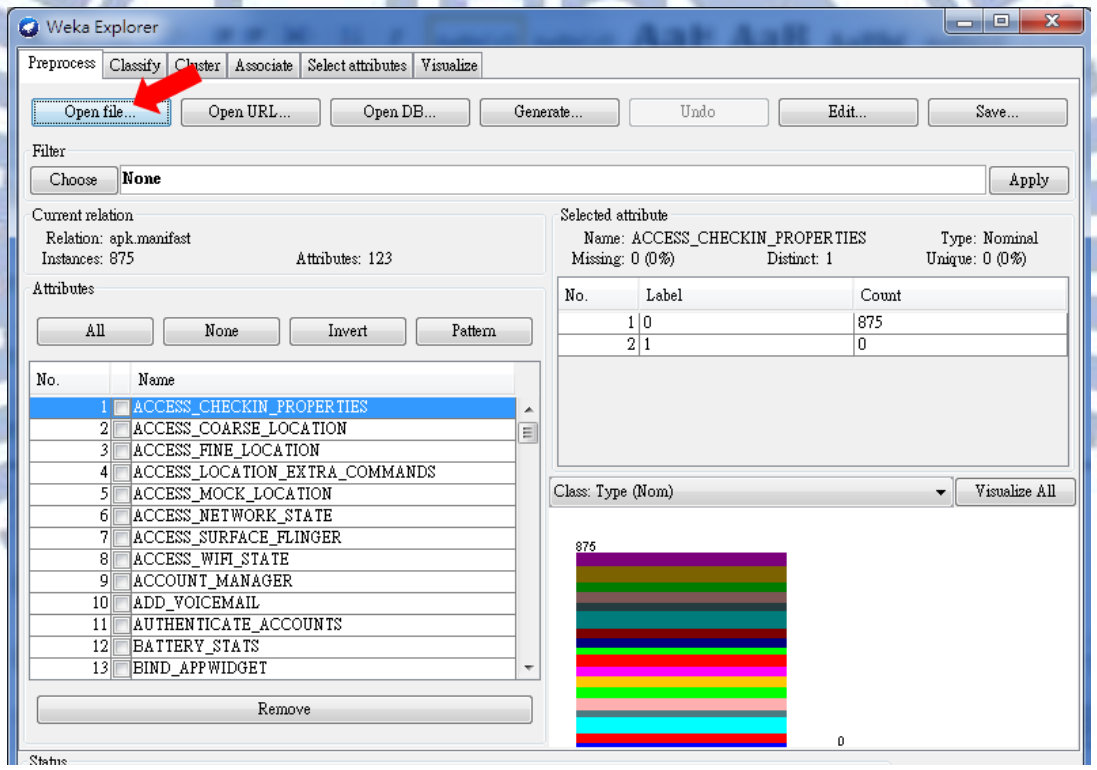


圖 18 Weka 操作介面 1

在這裡我們可以移除不要的 attribute(例如說這次我們對 app 自訂的 permission 不分析，則可以把它移除)。

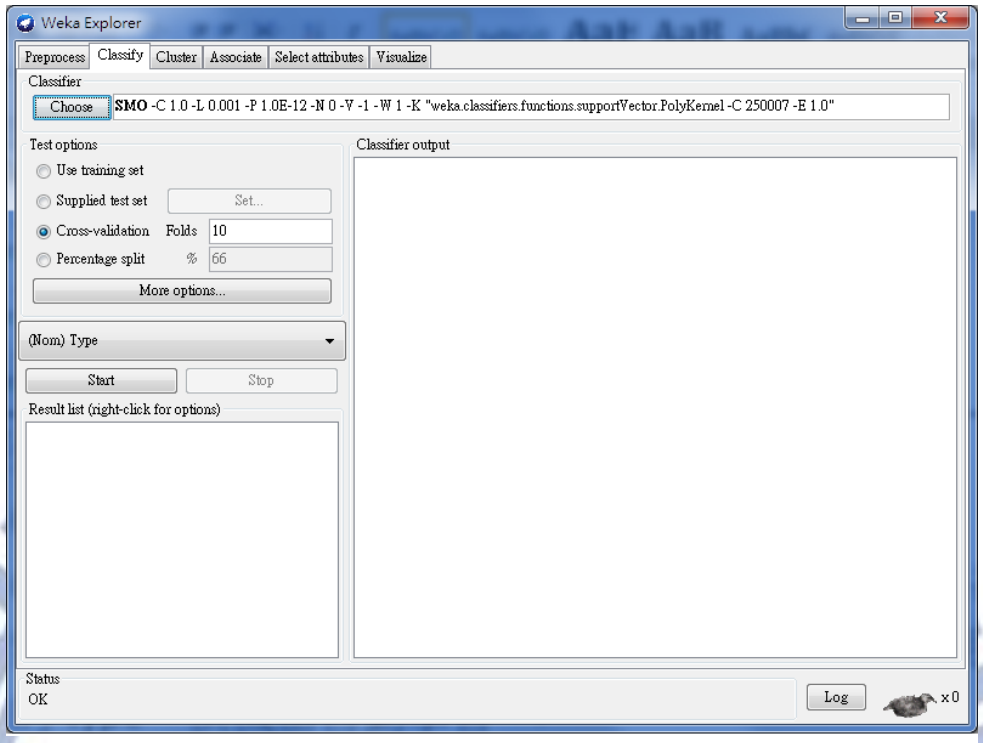


圖 19 Weka 操作介面 2

3. 再來選擇 Classify 標籤，Classifier 要選擇 weka>classifiers>functions>SMO
4. 在 SMO 指令列左鍵點一下會跳出設定選項，要修改的是 filter type，要改成 No normalization/standardization。
5. 設定完後按下 Start 就可以開始分析，結果會在右邊的 Output 欄。
6. 最後在 Result List 右鍵選擇 Save result buffer 就可以將結果存成檔案。

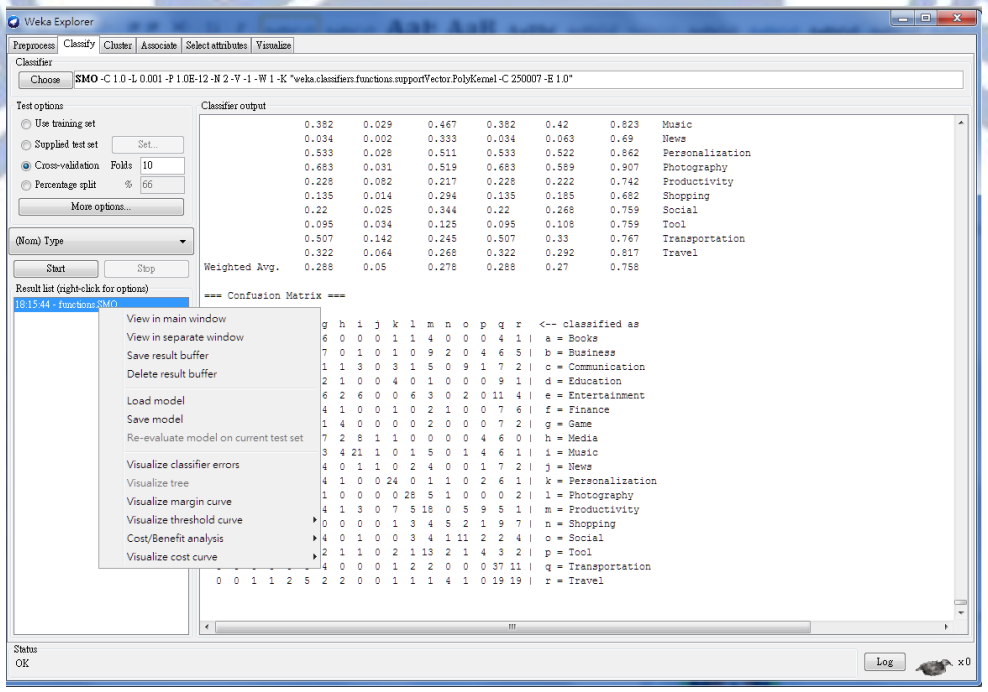


圖 20 Weka 操作介面 3

第五章、成果介紹

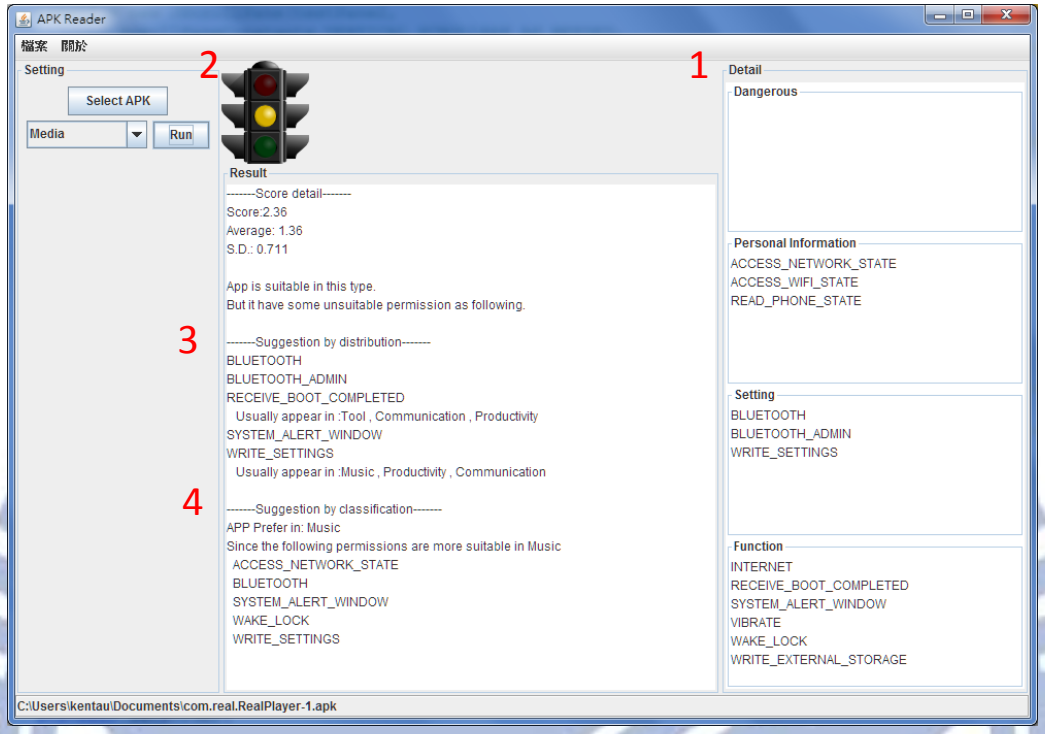


圖 21 實作介面

5.1 程式介紹

上面是我們實做出來的程式介面，分為以下三部份，並且說明如下：

5.1.1 permission 分類

這部分是由我們將 122 個 permission 依照功能分成四類：

Dangerous 是不該出現在一般 app 的 permission，此類 permission 功能可能造成使用者手機重大危害。

Personal Information 是使用者的個人資料存取權限的 permission，使用者可以檢視這個 app 會使用到的資料決定是否要安裝這個 app。

Setting 是有關於手機設定的部分，這些 permission 將會更改手機的設定，可能會造成耗電或是使用者的不便，使用者可以檢視這些 permission 來瞭解這個 app 會對手機設定造成什麼影響。

Function 是有關於手機功能的利用，這些 permission 會使用到手機上的資源，有些功能可能會造成手機耗電或是產生額外的費用，使用者可以從這裡知道 app 會用到什麼手機的功能。

藉由分類以及瀏覽 app 所需的 permission，使用者可以瞭解安裝此 app 後

可能會洩漏的資訊以及被利用的功能，藉由我們的分類，使用者可以更清楚而且關心重要資訊的安全性。

5.1.2 app 危險指數計算

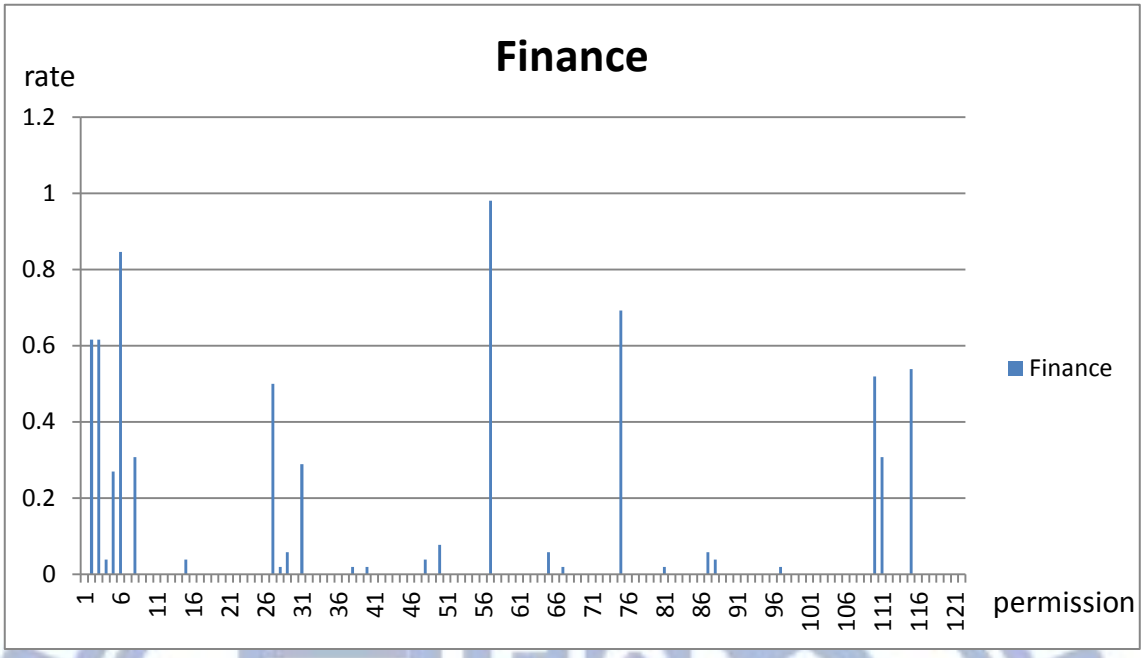
我們將 app 利用前面所敘述的方法評量危險指數，較高的危險指數代表所利用到的 permission 比較不尋常或是利用很多的資源。我們利用我們所蒐集的資料庫來做一個評量，Index 是此 app 的危險指數，Average 是此類型中所有 app 危險指數的平均值，S.D.是此類型中所有 app 危險指數的標準差，當輸入的 app 的危險指數高於平均兩個標準差，我們會提示為危險的 app，並且以紅燈表示，提醒使用者將要特別注意此 app 的來源以及他所使用到的 permission，他利用到的 permission 在這個 type 中相當的不尋常，可能是分類錯誤或是惡意的軟體。當輸入的 app 的危險指數高於平均一個標準差、小於兩個標準差時，我們將顯示為黃燈，提醒使用者這個 app 可能有利用到少數特殊的 permission，要優先注意這些特殊的 permission。當輸入的 app 的 index 未高於則顯示為綠燈，表示為安全的 app。

5.1.3 permission 與類別

當使用者對此 app 的危險程度大致了解後，我們會提供使用者需要注意的 permission 列表。第一種方式是利用 permission 在資料庫中的分佈特性。我們統計每個 permission 在每個 type 中的使用率，並且設定一個 threshold，當輸入 app 使用到的 permission 在輸入 type 中的使用率低於 threshold，我們會提醒使用者這個 permission 不太尋常，需要使用者特別注意。如果此 permission 在其他 type 的使用率高於 threshold，我們也會告知使用者，讓使用者瞭解這個 permission 在其他類別有被使用，或許這個 app 有其他類別的特性或功能。

使用者藉由這些資訊可以注意特別不尋常的 permission，而不用花心思把所有所需要的 permission 內容全部看完，也可以藉由我們的分析來判斷是否要提供這個 permission 給 app。

第二種方式我們利用資料分類的方法來給使用者建議，我們藉由和其他 type 比較，告知使用者那些 permission 需要被注意。我們會建議使用者一個 type，這個 type 是我們藉由資料分析的結果產生的，然後告知使用者分析的結果中那些 permission 是比較屬於建議的 type，這些 permission 比較適合出現在我們建議的 type，使用者可以判斷 app 是否是這個類別的程式，如果不是則需特別注意這些 permission 是否為誤用或是惡意的程式。



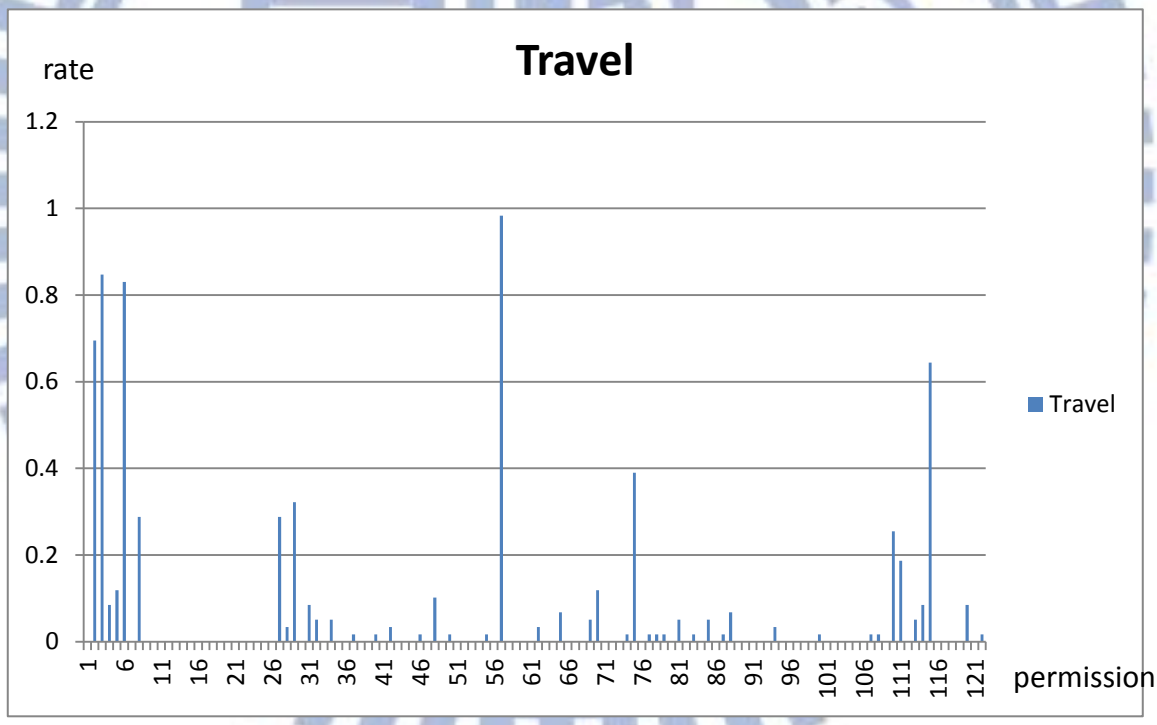
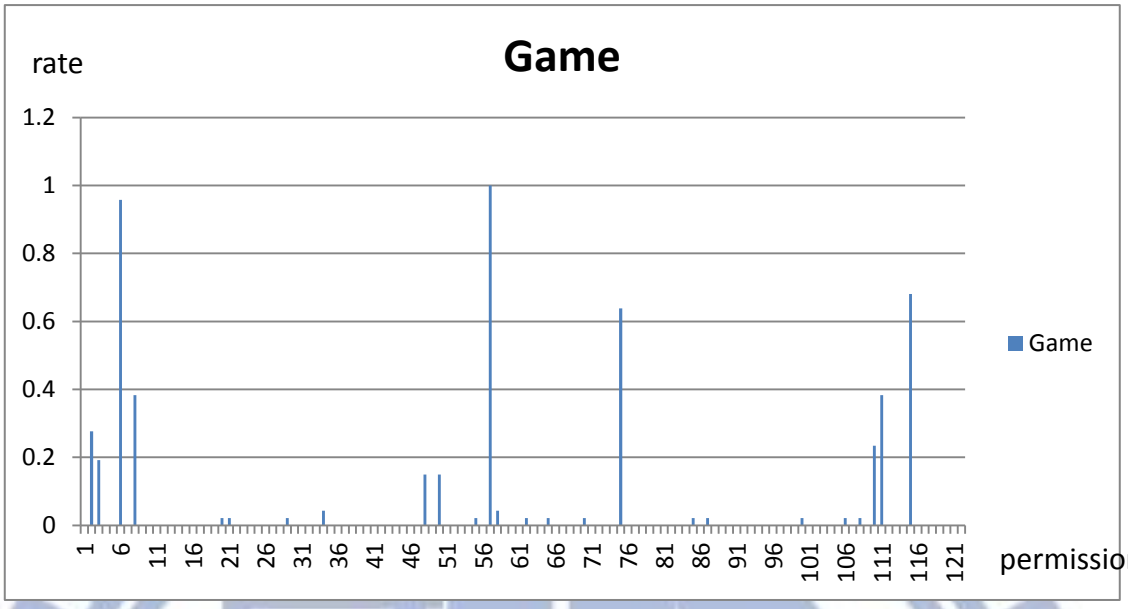


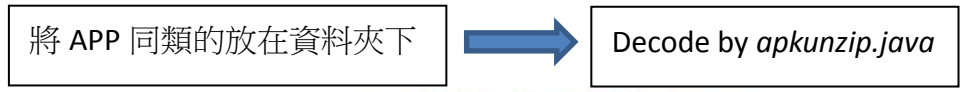
圖 22 Permission 使用率圖



這裡我們舉出三類的 Permission 使用率圖來介紹。我們發現在同一類的 app 當中會使用到的 permission 是分佈不均的，這和 Barrera 發表的 paper[7]，結論相符，而我們則可以利用這些統計資料來幫助使用者分析 app 的 permission。

5.2 程式流程

5.2.1 Preprocess



apkunzip.java：

Input： dictionary path

Output： decode all app to dictionary/unpack

UseSaxParse.java

Input： unpack path

Output： database file

ReadDatabase.java

Input： database path

Output： permission rate(setting) 、 type index (setting2)

利用 Weka 產生出結果檔案，提供給後面分析利用。

實作程式(*ReadInputUI.java*)

5.2.2 分析

分析 1：

將讀取出來的 permission 依照下面分類輸出：

表 3 Permission 分類

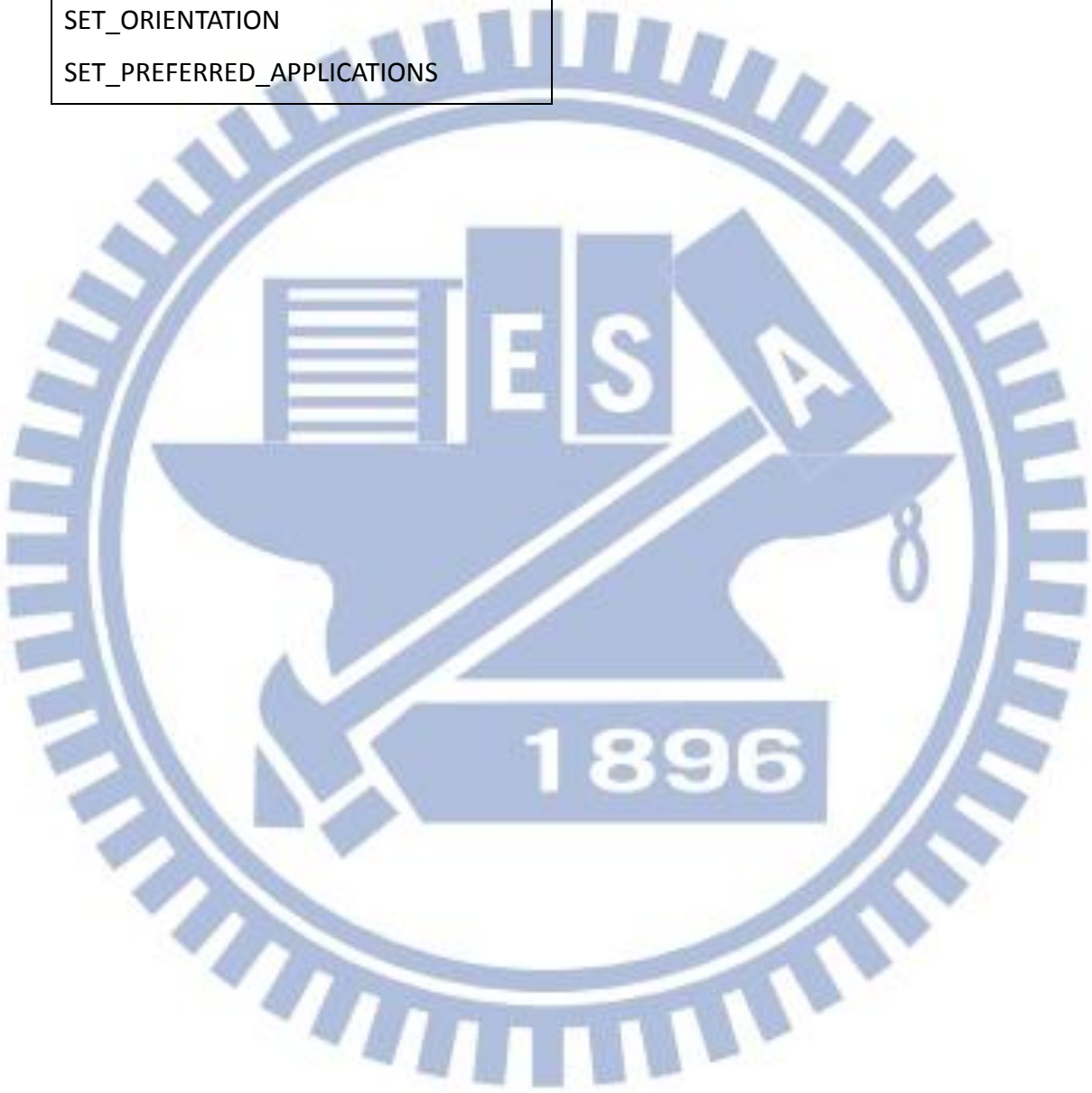
Personal Information
ACCESS_CHECKIN_PROPERTIES
ACCESS_COARSE_LOCATION
ACCESS_FINE_LOCATION
ACCESS_LOCATION_EXTRA_COMMAND
ACCESS_MOCK_LOCATION
ACCESS_NETWORK_STATE
ACCESS_WIFI_STATE
ACCOUNT_MANAGER
AUTHENTICATE_ACCOUNTS
GET_ACCOUNTS
GET_PACKAGE_SIZE
GET_TASKS
MANAGE_ACCOUNTS
READ_CALENDAR
READ_CALL_LOG
READ_CONTACTS
READ_HISTORY_BOOKMARKS
READ_LOGS
READ_PHONE_STATE
READ_PROFILE
READ_SMS
SUBSCRIBED_FEEDS_READ
SUBSCRIBED_FEEDS_WRITE
USE_CREDENTIALS
Setting
BIND_APPWIDGET
BIND_DEVICE_ADMIN
BIND_INPUT_METHOD
BIND_REMOTEVIEWS

BIND_TEXT_SERVICE
BIND_VPN_SERVICE
BIND_WALLPAPER
BLUETOOTH
BLUETOOTH_ADMIN
CHANGE_COMPONENT_ENABLED_STAT E
CHANGE_CONFIGURATION
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
DEVICE_POWER
DIAGNOSTIC
DISABLE_KEYGUARD
DUMP
EXPAND_STATUS_BAR
FORCE_BACK
INJECT_EVENTS
INSTALL_LOCATION_PROVIDER
INSTALL_PACKAGES
INTERNAL_SYSTEM_WINDOW
MODIFY_AUDIO_SETTINGS
MODIFY_PHONE_STATE
MOUNT_UNMOUNT_FILESYSTEMS
SET_ACTIVITY_WATCHER
SET_ALWAYS_FINISH
SET_ANIMATION_SCALE
SET_DEBUG_APP
SET_POINTER_SPEED
SET_PROCESS_LIMIT

SET_TIME
SET_TIME_ZONE
SET_WALLPAPER
SET_WALLPAPER_HINTS
SIGNAL_PERSISTENT_PROCESSES
WRITE_APN_SETTINGS
WRITE_SECURE_SETTINGS
WRITE_SETTINGS
Function
ADD_VOICEMAIL
BATTERY_STATS
BROADCAST_PACKAGE_REMOVED
BROADCAST_SMS
BROADCAST_STICKY
BROADCAST_WAP_PUSH
CALL_PHONE
CALL_PRIVILEGED
CAMERA
CONTROL_LOCATION_UPDATES
FLASHLIGHT
GLOBAL_SEARCH
INTERNET
NFC
PROCESS_OUTGOING_CALLS
READ_FRAME_BUFFER
READ_SYNC_SETTINGS
READ_SYNC_STATS
RECEIVE_BOOT_COMPLETED
RECEIVE_MMS
RECEIVE_SMS
RECEIVE_WAP_PUSH

RECORD_AUDIO
REORDER_TASKS
SEND_SMS
SET_ALARM
STATUS_BAR
SYSTEM_ALERT_WINDOW
UPDATE_DEVICE_STATS
USE_SIP
VIBRATE
WRITE_CALENDAR
WRITE_CALL_LOG
WRITE_CONTACTS
WRITE_EXTERNAL_STORAGE
WAKE_LOCK
WRITE_GSERVICES
WRITE_HISTORY_BOOKMARKS
WRITE_PROFILE
WRITE_SMS
WRITE_SYNC_SETTINGS
WRITE_USER_DICTIONARY
Dangerous
BRICK
CLEAR_APP_CACHE
CLEAR_APP_USER_DATA
DELETE_CACHE_FILES
DELETE_PACKAGES
FACTORY_TEST
HARDWARE_TEST
KILL_BACKGROUND_PROCESSES
MANAGE_APP_TOKENS
MASTER_CLEAR

MOUNT_FORMAT_FILESYSTEMS
PERSISTENT_ACTIVITY
READ_INPUT_STATE
REBOOT
RESTART_PACKAGES
SET_ORIENTATION
SET_PREFERRED_APPLICATIONS



下面是有關 Dangerous Permission 的說明，這些 permission 一部分是危險功能，另一部分是因為系統修改已經移除，不該出現在一般 app 當中：

BRICK

此 permission 可能造成裝置無法使用，或是關閉手機功能。這是非常危險的 permission，不該出現在平常的 app 中。

CLEAR_APP_CACHE

允許 app 清除所有 app 的 cache 資料。若是惡意 app 使用，可能造成個人資料損毀或遺失。

CLEAR_APP_USER_DATA

允許 app 清除所有使用者的資料。若是惡意 app 使用，可能造成個人資料損毀或遺失。

DELETE_CACHE_FILES

允許 app 刪除所有 cache 資料。若是惡意 app 使用，可能造成個人資料損毀或遺失。

DELETE_PACKAGES

允許 app 刪除其他的 app。若是惡意 app 使用，可能造成已安裝的程式損毀或是刪除。

FACTORY_TEST

讓 app 可以在工程模式下測試，取得系統權限。若是惡意 app 使用，可能造成系統資料損毀。

HARDWARE_TEST

讓 app 可以在進入系統硬體中。若是惡意 app 使用，可能造成系統資料損毀。

KILL_BACKGROUND_PROCESSES

允許 app 關閉背景執行程式。若是惡意 app 使用，可能造成系統損毀，破壞系統程式。

MANAGE_APP_TOKENS

允許 app 管理(創建、刪除)程式 token。若是惡意 app 使用，可能造成系統損毀，破壞系統程式。

MOUNT_FORMAT_FILESYSTEMS

允許 app 掛載，卸除系統資料。若是惡意 app 使用，可能造成系統損毀，破壞系統程式，或是洩漏個人檔案。

PERSISTENT_ACTIVITY

允許 app 一直在前螢幕執行。此 permission 可能造成系統當機或是過度消耗電源。已經在新的 API 取消支援。

READ_INPUT_STATE

此 permission 已經在新的 API 取消支援。

REBOOT

允許 app 重新啟動裝置。若是惡意 app 使用，可能造成系統當機或毀損系統。

RESTART_PACKAGES

此 permission 已經在新的 API 取消支援。

SET_ORIENTATION

重新設定手機感應位置，允許進入 low-level 硬體設定。若是惡意 app 使用，可能造成系統當機或毀損系統。

SET_PREFERRED_APPLICATIONS

允許 app 修改 process 列表參數。若是惡意 app 使用，可能造成系統當機或毀損系統。

分析 2：

讀取以下資料

1. App permission

2. Permission rate(setting)

3. Type index (setting2)

利用 1.和 2. 計算 app index，並且依照資料 3 來提供建議。

計算方法：Index (P , t)=$\sqrt{\sum_{p\in P}(1-R(p,t))^2}$，資料 1 得到 P，資料 2 提供$R(p,t)$。

再由資料 3 得出每個 type 的平均值(Avg.)和標準差(SD)，app 的 index 大於平均值兩個標準差則顯示紅燈，大於平均值一~二個標準差間則顯示黃燈，其餘為綠燈，並且顯示各種燈號對應的意義。

下面是資料庫中，依照以上標準 app 在各類中所佔的百分比。

Type	>Avg.	>(Avg+SD)	>(Avg+2*SD)	Type	>Avg.	>(Avg+SD)	>(Avg+2*SD)
Books	52.63%	21.05%	0.00%	News	65.52%	10.34%	0.00%
Business	45.65%	17.39%	4.35%	Personalization	46.67%	15.56%	2.22%
Communication	46.05%	17.11%	2.63%	Photography	39.02%	24.39%	0.00%
Education	48.28%	20.69%	0.00%	Productivity	46.84%	16.46%	3.80%
Entertainment	51.85%	14.81%	1.85%	Shopping	43.24%	18.92%	0.00%
Finance	50.00%	21.15%	0.00%	Social	54.00%	22.00%	0.00%
Game	40.43%	17.02%	2.13%	Tool	42.86%	19.05%	4.76%
Media	54.76%	14.29%	2.38%	Transportation	39.73%	13.70%	4.11%
Music	47.27%	12.73%	5.45%	Travel	49.15%	18.64%	1.69%

表 4 Database index 列表

因為不同 type 所會用到的 permission 數量不同，所以我們對於每個 type 採用不同的標準，由同一類中的平均值作為比較，經由這樣的標準得到的結果可以使得各類別間 app 的分佈較為一致。

分析 3：

讀取以下資料

1. App permission

2. Permission rate(setting)

列出使用率低於 threshold 的 permission，並且尋找此 permission 是否有在其他 type 中有超過 threshold 的使用率。

分析 4：

讀取以下資料

1. App permission

2. Classification result

利用 data mining 方法，檢視 app 所需的 permission，是否應屬於其他 type。

讀取使用者所選的 app type 與其他 type 間互相分類的結果(資料 2)，然後將 app 使用的 permission(資料 1)帶入做比較，判斷算出的結果是否與使用者選取的 type 相同。找出不同的 type，並且讀取結果判斷哪些 permission 要回報給使用者。

1. 讀取 classification result 中，使用者所選的 type 和其他 type 的分類結果。
2. 每一對分類結果將 permission 帶入，算出結果是否符合使用者所選。
   - 符合則 pass
   - 不符合檢視是否有使用到不屬於使用者所選 type 的 permission，沒有使用則 pass，有使用則將此結果計入 candidate。
3. 如果只有一組 candidate 則輸出結果。

如果超過一組則依照下列方法投票決定出一組最佳結果 output。

I. 這裡我們採用 vote 機制選出最佳的結果。

II. 我們將 candidate set 中每兩個 candidate 一組做比較，利用 Classification result 中的結果，比較中獲勝的一方可以得到一票，最後統計出最高得票的，當成 output 的結果。

4. 將結果和具有此結果特性的 permission 回報給使用者。

5.3 分析實例

5.3.1 安全 app (Monkey Jump 2)

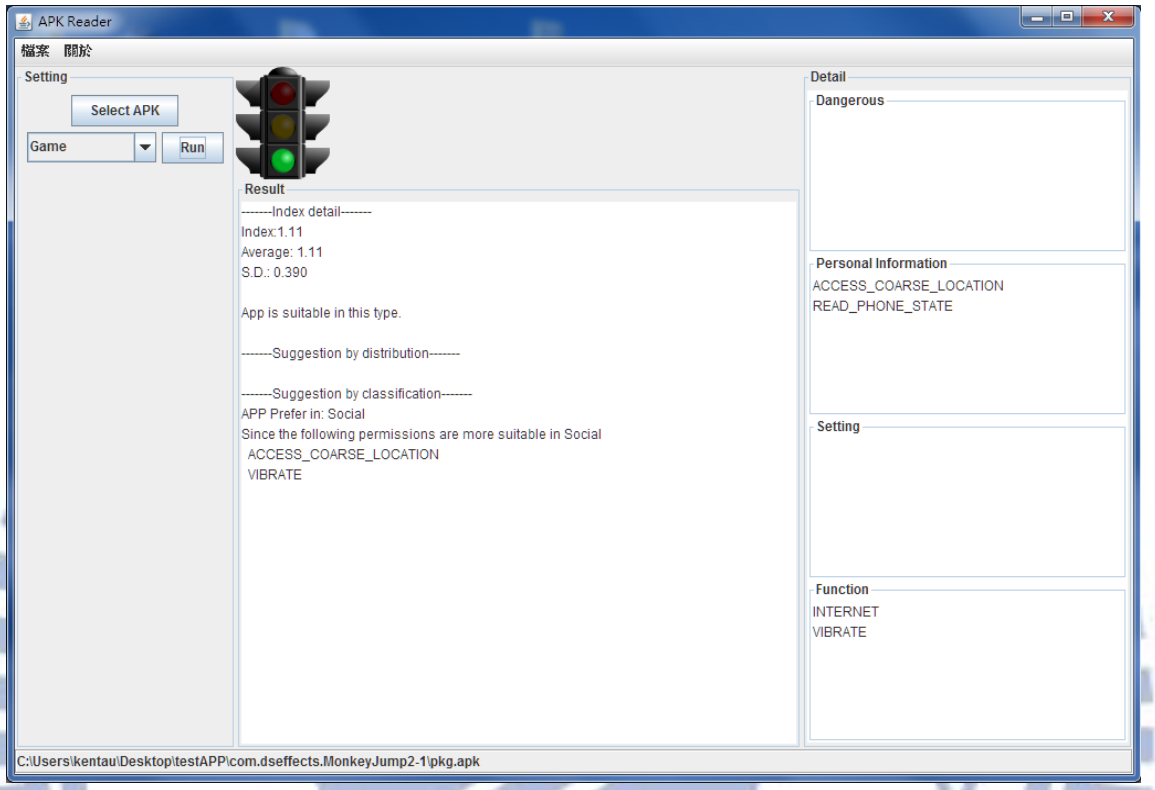


圖 23 分析實例 1

這是一個由 Google Play 下載的 app：Monkey Jump 2，得到了它所需的 permission 以及我們選擇他的類別是 Game，上面是程式跑出來的結果。

一開始我們可以簡單的看出來這個 app 是綠燈，也就是在分析結果中我們認為他是安全的。詳細的來看下面分析結果，因為這個 app 使用的 permission 並不多，所以他的危險指數在 game 的類別中沒有超過平均值，我們顯示這個 app 在這個別中是合理的，從 permission distribution 的報告我們發現他並沒有使用到低使用率的 permission，但是由 permission classification 的報告我們發現他因為有使用到讀取所在位置(ACCESS_COARSE_LOCATIOM)以及震動(VIBRATE)這兩個 permission，所以可能這個 app 也具有 Social 這個類別的特性。

最後右邊我們可以去一一看這個 app 所用到的手機資源。第一他並沒有使用到我們列為 Dangerous 的 permission，安裝這個程式可能會洩漏的個人資訊有位置以及手機狀態，這支 app 不會修改到系統設定以及它會用到網路和震動這兩個手機的功能。

5.3.2 危險 app (Monkey Jump 2 infected)

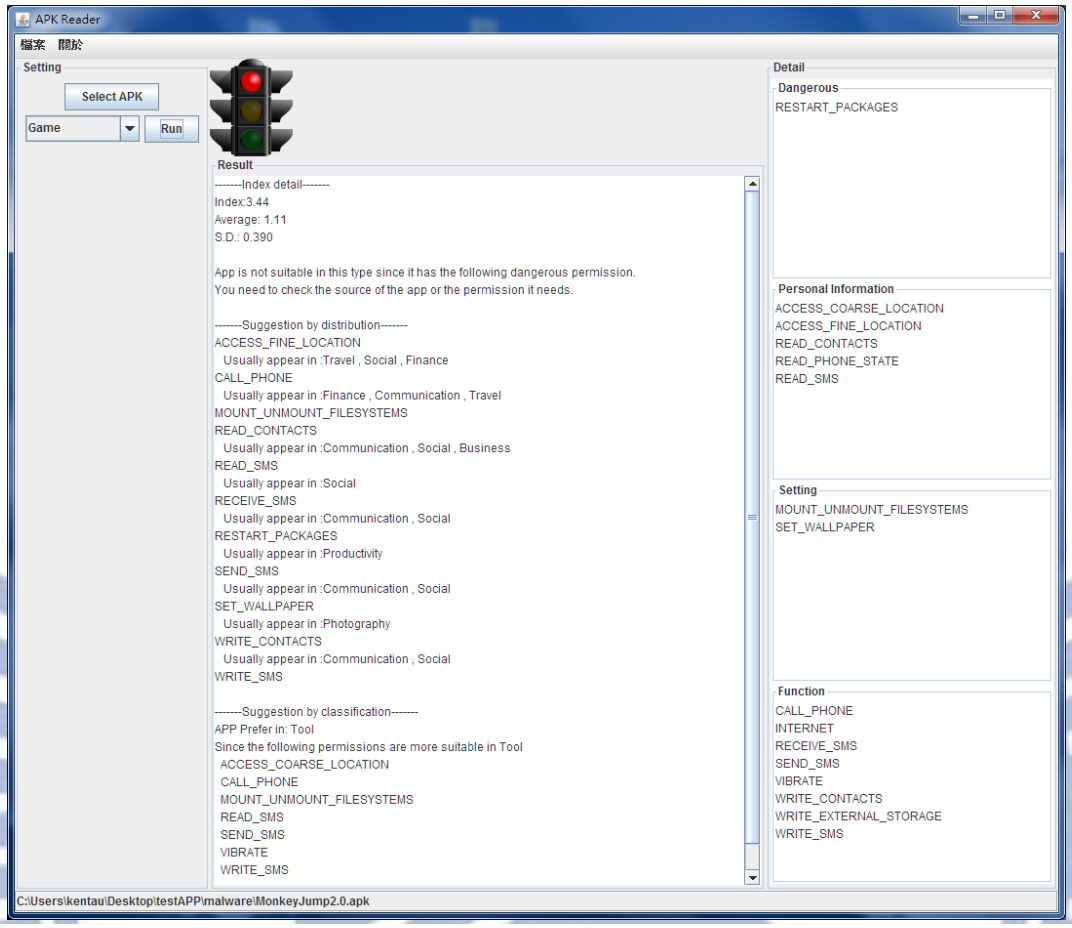


圖 24 分析實例 2

這是一個我們由網路上論壇下載的 app：Monkey Jump 2，和上面例子不同的是這是支經過惡意修改，受到 Geinimi 木馬程式感染的遊戲。得到了它所需的 permission 以及我們選擇他的類別是 game，上面是程式跑出來的結果。

一開始我們可以簡單的看出來這個 app 是紅燈，也就是在分析結果中我們認為他是危險的。

我們也提供詳細的說明在下方，由於這個 app 使用到非常多的權限，這在遊戲類當中是不合理的，所以他的危險指數跟平均值來說會高出許多，超過了兩個標準差我們即判定他為危險。

經過木馬程式感染後的 app 要求了相當多的權限來蒐集使用者的個人資料，他讀取了包括 SMS，CONTACTS 等等的資訊，這些 permission 在同類當中是相當少出現的，我們在報告中一一列出來提醒使用者這些 permission 不適合出現在遊戲類。舉個例子來說，他有用到讀取以及發送 SMS 的功能，這應該是屬於

communication 類的，我們就在這邊提醒使用者，除非這個 app 有 communication 的功能，否則這些 permission 是不適當的。

從 permission classification 的報告中我們判斷它應該是一個 tool 類別的程式，但是這支 app 宣稱是一個遊戲程式，所以他可能除了遊戲的功能外，還具有一些 tool 類的功能，下面顯示他蒐集了個人資料，播打電話等等功能是 tool 類的特徵，當使用者看到這個報告後就可以知道它除了遊戲功能外還有做這些額外的功能。

最後右邊我們可以去一一看這個 app 所用到的手機資源。第一他有用到危險的 permission，如果是惡意程式的話可能導致系統當機及損害。在個人資訊的方面他使用到了五項的資訊，這個程式也會更改系統的設定以及使用了多達八項的手機功能。

5.3.3 分類錯誤 app (Realplayer)

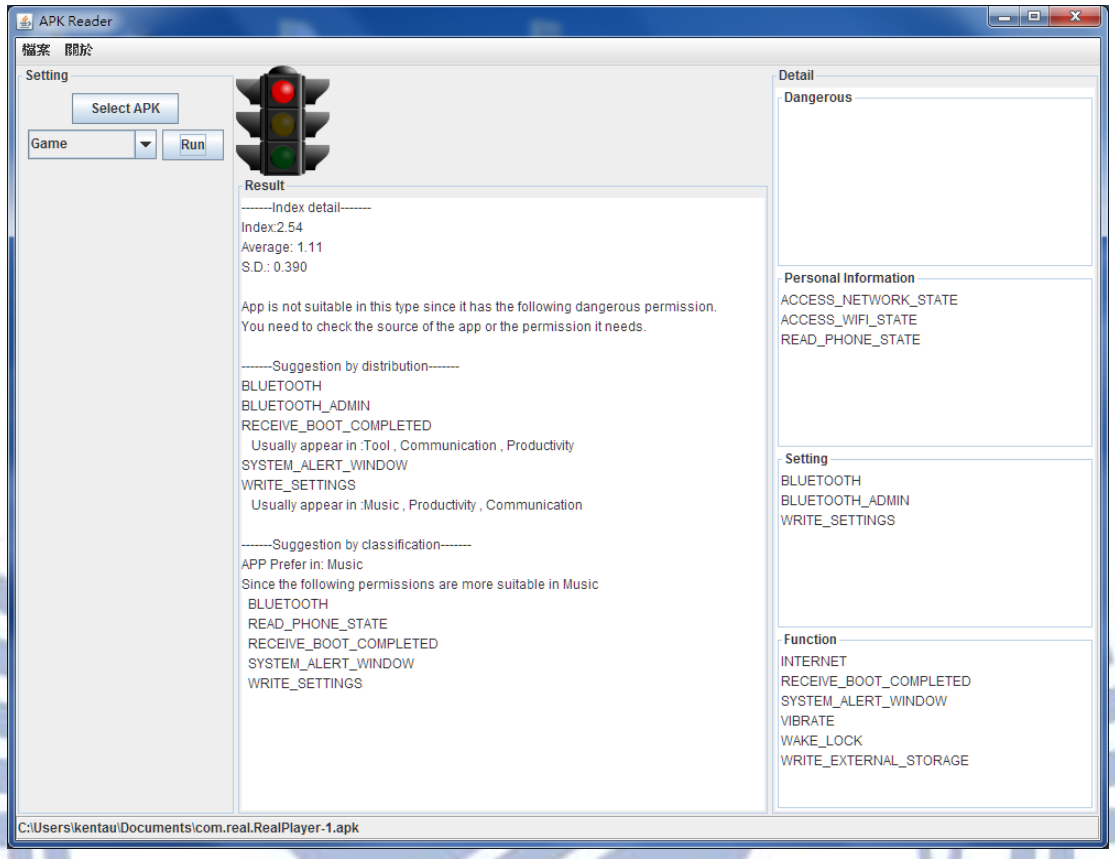


圖 25 分析實例 3

最後我們展示一個 app 分類錯誤的例子。這是一個由 Google Play 下載的播放器 Real Player。如果我們當成 Game 類別的 app 來看，會發現他利用到的很多 permission 是不該出現在 Game 中的，但是我們可以藉由 permission classification 的報告中發現這應該是一個 Music 的程式，於是經過我們建議使用者再度查詢這個程式的資料就能發現是錯誤選擇了它的類別，可以讓使用者再一次確定 app 的功能和內容，確定後再度檢查就會得到正常的結果。

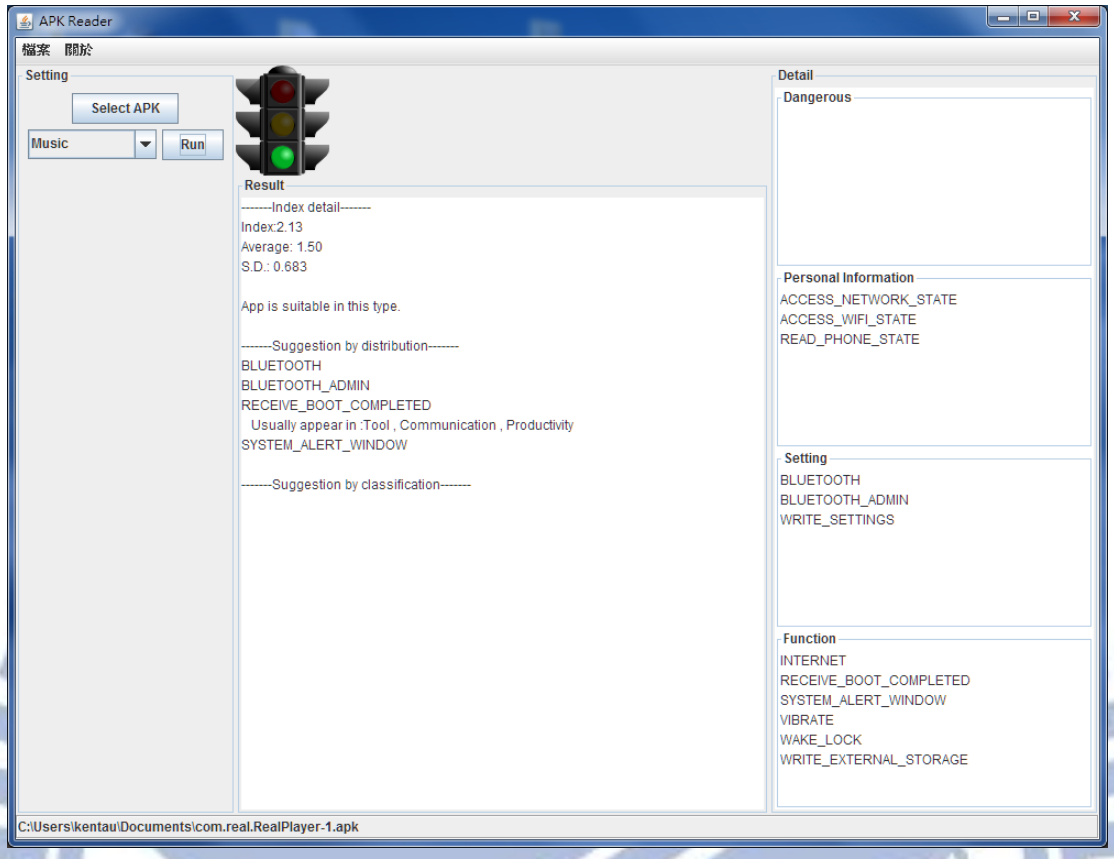


圖 26 分析實例 4

Real Player 在 Music 中則為適合的分類，但他還是有用到一些比較少見的 permission，我們在報告中還是會提醒使用者注意這些資訊。

第六章、結論和未來工作

隨著智慧型手機的銷售數量日益增加，雖然目前尚未成為惡意軟體的首要攻擊目標，但仍有跡象顯示手持裝置資訊安全的未來重要性。

以往的研究多專注在分析程式碼，或是修改系統來增加安全性。在這邊我們提出一個可以簡單使用，不需要修改系統就能得到清楚報告程式。使用者可以藉由此程式，省去一一閱讀 permission 的麻煩，又可以清楚的了解 app 所得到的資料和系統資源，而非盲目的下載和安裝 app。我們提供了三個面向來幫助使用者了解 app：

1. permission 分類
2. 程式危險指數(安全性)計算，以及由 permission 分布提供建議
3. 由資料分類提供建議

藉由蒐集 app 資料以及建立資料庫，可以讓 app 和其他的 app 之間做比較，而非死板的定義危險 permission。

在未來我們希望可以加入和使用者互動的方法：

1. 使用者提供的 permission 和 type 將不再只是測試，也可以加入到我們蒐集 app 的資料庫中，進一步的去更新我們的資料庫。
2. 對使用者加入客製化功能，使用者可以選擇 permission 開放，或是調整 risk index 的權重，對於較不重視的 permission 降低權重。

我們也希望能夠對蒐集到的 app 做進一步的分析，一方面去更新我們的資料庫(例如有宣告 permission 但在 app 中並未使用)，或是找出更多的方法來檢測惡意軟體。

另外希望加入使用者上傳 app 的機制。因為許多大型 app 會用到很多手機功能，宣告許多 permission，我們程式可能會判斷為危險的 app，我們可以藉由讀取 AndroidManifest.xml 中的 package 參數或是由 app 的名字中讀取出 app 的發行商作為判斷，將一些安全的公司加入其中建立白名單(white list)，並且加驗 app 的簽章確認正確性。

參考文獻

[1] nielsen, http://www.nielsen.com/global/en.html

[2] Distimo, http://www.distimo.com/

[3] Takamasa Isohara, Keisuke Takemori, and Ayumu Kubota "Kernel-based Behavior Analysis for Android Malware Detection," Seventh International Conference on Computational Intelligence and Security (CIS), 2011 ,pp. 1011-1015

[4] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth "TaintDroid: an information-flowtracking system for realtime privacy monitoring on smartphones," in Proceeding of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2010, pp. 393-408

[5] Farrukh Shahzad, Sohail Bhatti, Muhammad Shahzad, and Muddassar Farooq "In-Execution Malware Detection using Task Structures of Linux Processes," in Proceedings of the IEEE International Conference on Communication (ICC), 2011, pp. 1-6

[6] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri "A study of android application security," in Proceedings of the 20th USENIX conference on Security, 2011, pp. 21-21

[7] David Barrera, H. Güneş Kayacik, Paul C. van Oorschot, and Anil Somayaji "A methodology for empirical analysis of permission-based security models and its application to Android," in Proceedings of the 17th ACM conference on Computer and communications security (CCS), 2010, pp. 73-84

[8] William Enck, Machigar Ongtang, and Patrick McDaniel "On lightweight mobile phone application certification," in Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), 2009, pp. 235-245

[9] Mohammad Nauman, Sohail Khan, and Xinwen Zhang "Apex: Extending Android permission model and enforcement with user-defined runtime constraints," in Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, 2010, pp. 328–332

[10] Leonid Batyuk, Markus Herpich, Seyit Ahmet Camtepe, Karsten Raddatz, Aubrey-Derrick Schmidt, and Sahin Albayrak "Using Static Analysis for Automatic Assessment and Mitigation of Unwanted and Malicious Activities Within Android Applications," in Proceedings of the 6th International Conference on Malicious and Unwanted Software (MALWARE), 2011, pp.66-72

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik "A training algorithm for optimal margin classifiers," in Proceedings of the 5th Annual Workshop on Computational Learning Theory, ACM , 1992, pp. 144–152.