適用於 Android 應用程式的隱私風險評估機制

葉國暉(Kuo-Hui Yeh)[1]，郭仁宗(Ren-Zong Kuo)[2]，廖皓翔(Hao-Xiang Liao)[3]，楊芳捷(Fang-Jie Yang)[4]，林秉賢(Ping-Hsien Lin)[5]，林子鈞(Tzu-Chun Lin)[6]
[134] 國立東華大學資訊管理系，花蓮，台灣
khyeh@mail.ndhu.edu.tw[1]；610139006@ems.ndhu.edu.tw[2]；summerfeel27@gmail.com[3]
[2] 慈濟技術學院資訊科技與管理系，花蓮，台灣
kuorezn@gmail.com[2]
[56] 國立台灣科技大學資訊管理系，台北，台灣
m10209104@mail.ntust.edu.tw[4]；m10309116@mail.ntust.edu.tw[5]

摘要

近來，Android 系統上的行動應用服務逐漸普及在人們的日常生活之中，然而，在這些行動服務帶來便利性的同時，越來越多的個人敏感性資料皆被有意且無意地儲存在 Android 系統之中，如此將使得駭客有較高的機會來拼湊出個人(或組織)的隱私。鑑於此，本研究將針對 Android 系統上的隱私管理進行探討與研究，文中主要提出了一套適用於 Android 系統上的應用程式隱私分析框架，稱為 *AppLeak*，用以對 Android 應用程式進行資訊損失評估、隱私洩漏檢測和隱私風險評估。於實作面，*AppLeak* 系統採用使用者感知和客觀攻擊意識等新興觀點，用以針對 Android 系統上的 Facebook 與 LINE 兩套行動軟體進行隱私分析，作為本研究框架之可行性測試。

關鍵詞：Android 應用程式、安全、隱私、風險評估

壹、前言

近期，對於智慧型手機技術的研究迅速發展，讓其成為了產、官、學的聚光燈焦點。許多企業也因此開始聚焦於智慧型手機的行動應用與服務開發。在眾多的行動式作業系統中，Android 是最受歡迎的系統架構之一，居高不下的市占率(將近八十個百分比)帶動著 Android 應用程式與加值服務的成功發展，並帶來在系統與應用程式開發之外的廣大附加價值。通常，消費者可以從線上市集(如：Google Play、App Store、Windows Marketplace 與 Nokia OVI Store)下載行動版應用程式。透過這些市集，數以萬計的行動應用程式可以被正確地下載並安裝至使用者的智慧型手機中，為企業、政府與個人提供高價值的服務。隨著行動服務的多樣性、彈性與使用效率，Android 應用程式的開發在近幾年變得越來越熱門。

在智慧型手機提供著人們越來越多的生活便利與大眾娛樂之同時，也帶來了更多資訊安全與個人隱私的潛在威脅。例如：每一個智慧型手機都擁有一獨特的國際移動設備識別碼(IMEI；International Mobile Equipment Identity)，是一組可以識別智慧型手機的15 位數識別號碼。一旦 IMEI 碼被竊取，偽裝成使用者的行為就會成為可能。此外，病毒、蠕蟲和木馬等傳統的安全弱點也同樣是行動裝置的潛在威脅。另一方面，近年來資訊洩漏的問題也漸漸的浮上臺面，使得個人隱私洩漏偵測與分析也受到了許多注意，且隨著智慧型手機使用程度的增加，使用者隱私防護的意識也變得越來越重要。目前，大多數的行動應用程式都是由第三方開發並上傳至 Android 應用程式市集來達到宣傳的效果和額外的商業目標，如果沒有適當的審核機制，惡意的第三方開發者將可隨意地上傳其所開發的應用程式於市集中，並靜靜地等待人們下載。再者，人們使用應用程式時，第三方開發者可能會收集儲存在使用者的智慧型手機上之敏感資訊並透過網路散播。例如，應用程式可能會存取來自輸入裝置(如記憶體、相機和麥克風等)和感應器(如 GPS、加速偵測計和陀螺儀)個人機敏資訊，這個過程通常不需要使用者的授權，就可能會發生使用者的敏感性資訊洩漏。

在 Android 系統上，SQLite 一直被作為主要的儲存技術之一，因為一般智慧型手機的系統資源有限，傳統的關聯式資料庫並不完全適合作為主要的儲存技術，且儲存系統的需求也必須符合智慧型手機的架構。SQLite 擁有容易使用、強大、快速、耗用較少系統資源等優勢，非常適合使用於智慧型手機架構上的應用程式，因此，這幾年來可看見 SQLite 技術被廣泛地應用於智慧型手機系統之中[1]。在 2009 年，Junyan 等人[10]由 API 功能和內部系統架構兩個方面來分析嵌入式資料庫和 SQLite 的相異處，主要做法為在 ARM-Linux 模擬環境上實作 SQLite 和可以有效率的管理資料庫之呼叫介面函式，Junyan 等人的努力證明了 SQLite 可以優秀的符合嵌入式系統中的應用程式之資料庫需求。在 2010 年，Chen 等人[3]將植基於 SQLite 的資料庫管理系統移植到 C/OS-II(一款嵌入式即時作業系統)中，並透過應用程式驗證實作基礎功能的可能性，結果顯示了 SQLite 的強大資料庫管理功能具有平台獨立性與複雜資料的處理可行性。顯而易見的，SQLite 資料庫因其在嵌入式應用程式領域的特殊優勢，而成為 Android 架構上最受歡迎的儲存系統之一。

最近，Samarati 和 Sweeney[16][15][17]提出了可以廣義化的隱私定義。透過嚴謹的管理過程，來有效防制資訊洩漏的問題。在這之後，Biswas 等人[2]量化了低階系統的資訊存取對隱私的影響，並將此資訊轉為使用者可以瞭解的高階評分的隱私度量，且作者透過圖像化等使用者友善格式，提供了隱私察覺應用程式的架構和設計，讓使用者可以定義特定的隱私分析與防護政策。Ulltveit-Moe 等人[18]提出了在重要的資訊基礎設施中的隱私控制方法論，提出的方法論可以用來辨識、量化並降低隱私資訊的洩漏。之後，Kuzuno 等人[8]採用了一個基於 HTTP 封包目的地和內容距離(Content Distance)的分群方法，來建立用於偵測機敏資訊的特殊簽名，該系統不會修改 Android 框架的任何特定權限，而且可以簡單的實作於系統之中，用以管理可疑的應用程式之網路行為。根據以上

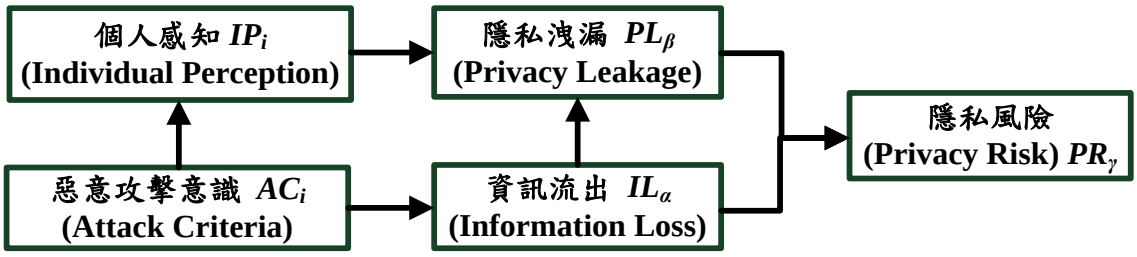
研究成果與趨勢發展，本研究了解到一套適用於 Android 應用程式的機敏資訊洩漏分析與隱私風險評估是十分重要的，但如何在既有的系統中找到不同的分析準則與資料分析手法則是最大的挑戰之一，因此本研究的主要研究目的將利用不同於以往研究的判定觀點來作為 Android 應用程式的機敏資訊洩漏分析與隱私風險評估之主軸，並用正式化表達來呈現該方法的精神。

貳、研究問題 – 行動應用程式下的隱私洩漏

一般來說，使用者通常會在不知道個人資料會如何被使用的情況下接受其欲安裝的應用程式之權限存取請求。例如，使用者可能在一款娛樂應用程式的安裝過程就允許其存取該使用者的部分(或所有)敏感資料之權限，但是該使用者根本不知道有多詳細的資料將被截取或如何被使用。在另一個例子中，即時訊息應用程式可以被用來進行社交工程攻擊。也就是說，惡意攻擊者可以試著利用社交工程手法來欺騙使用者的隱私資訊，如透過魚叉式網路釣魚(spear phishing)或點擊劫持(click-jacking)獲得信用卡資訊。因此，如果系統不提供適當的隱私保護機制，將可能會導致無法挽回的隱私洩漏。皆下來，我們模擬四種攻擊情境來展示隱私洩漏的威脅。

1. Alice 喜歡使用自己的智慧型手機上的社交應用程式來與好友交談，但 Alice 並不知道該即時傳輸軟體在傳輸訊息上並沒有加密。因此，惡意攻擊者 Eve 可以隨意地抓取所有傳輸訊息，並獲得 Alice 的隱私資訊(如名字、電話號碼、生日和圖片)。
2. Edward 習慣使用他的 iPad 在線上購物，然而，該線上市集並沒有為行動裝置提供適當的安全機制，如此將導致 Edward 的購物喜好可能被洩漏。
3. Ken 住在台北，但是每逢週一、週三與週五皆在內湖工作，而與週二與週四則到新竹進行支援，惡意攻擊者 Eve 可能會從肯恩手機上的 GPS 數值來定位他。
4. Sean 在他的智慧型手機上下載了一個與 NBA 有關的遊戲，該款應用程式需要一些個人資訊來註冊，所有資訊皆有加密。然而，由於西恩使用與 Facebook 帳戶相同的帳戶密碼，惡意攻擊者 Eve 可能會從西恩的 Facebook 頁面獲得西恩的帳號密碼，並從該款遊戲得到 Sean 的隱私資訊。

隱私洩漏的偵測和防護是個複雜的過程，因為隱私可以被洩漏出來的方式是非常地多樣化，如何幫助使用者以智慧且有效率的方式保護他們的機敏或隱私資訊將是十分重要的課題。因此，本研究旨在分析 Android 應用程式的隱私洩漏與風險評估。在歷來的文獻中，為了避免 Android 應用程式的資訊和隱私洩漏，已經有許多研究[4][9][11][14][19][21] 已經被實行。然而，截至目前為止，並沒有一份研究利用正式化表達來呈現 Android 應用程式的隱私洩漏與風險評估，在本研究中，我們呈現了一款分析框架 *AppLeak*，主要利用五個分析準則概念(如圖一)：惡意攻擊意識、個人感知、資訊流出偵測、隱私洩漏分析和隱私風險評估，來探測 Android 應用程式的機敏資訊和個人隱私資訊的洩漏。

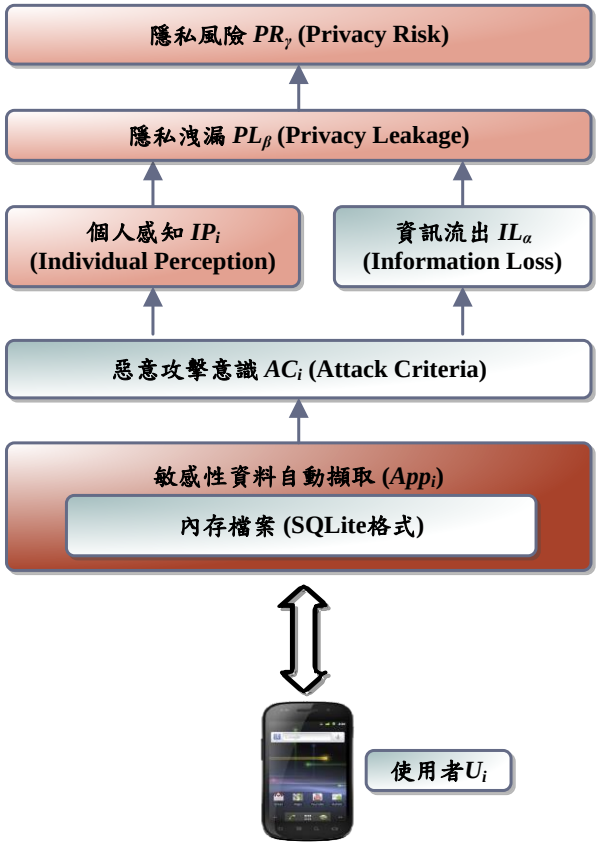


圖一：*AppLeak* 的分析準則與關聯性

在真實世界中，有惡意攻擊者會對許多使用者進行攻擊行為，如信用卡盜刷或社交工程攻擊，一旦成功，將可能造成使用者的大量損失。惡意攻擊的關鍵成功條件可能是固定的，因此，由使用者 U_i 操作的 Android 應用程式 App_i 的某特定攻擊之條件(或資訊)AC_i(Attack Criteria)將可事先被定義(在此統稱 AC_i 為惡意攻擊意識)。例如，信用卡盜刷的必要資訊有信用卡卡號、身份證字號、使用者生日、信用卡期限日期與卡片背面之三位數的驗證碼。此外，學過或是曾體驗過特定攻擊的人們，能更加體認讓攻擊發生的相關條件之重要性，這使得我們確信擁有信用卡盜刷知識的人們會對於保護信用卡號碼、信用卡期限日期和三位數驗證碼更有意識。因此，在 *AppLeak* 中，我們認為 AC_i 會影響使用者 U_i 對自身的資料或個人資訊之敏感性認知，故本研究推論惡意攻擊意識 AC_i 將會干涉使用者 U_i 對個人機敏資訊的感知度 IP_i(Individual Perception)，基於個人感知 IP_i，應用程式使用者可能會在他們的個人資訊之機敏度量上有些許不同。例如，Alice 可能會覺得將生日公開可能會導致潛在的網路詐騙威脅。而因為 Bob 期待會有大型的生日驚喜，這位年輕人總是希望大家提早知道他的生日。在這些例子中，這兩個人對於個人資訊的感知度上完全不同，因此使用完全相反的方式處理他的個人資訊。根據上述兩個觀點，本研究可更進一步地辨識 Android 應用程式 App_i 的機敏資訊流出程度 IL_α(Information Loss)與使用者隱私洩漏程度 PL_β(Privacy Leakage)。最後，隨著與 App_i 有關的資訊流出 IL_α 和隱私洩漏 PL_β 之分析結果，我們可以成功地評估 App_i 的隱私風險 PR_γ，並定義等級為低風險、中風險或高風險。

參、適用於 Android 應用程式的隱私風險評估框架：*AppLeak*

本節將針對 AppLeak 的系統架構進行說明，如圖二所示，由使用者 U_i 操作的 Android 應用程式 App_i 的 SQLite 內存檔案內容將自動被萃取出來作分析，而為了有效地評估 SQLite 萃取出來的資料是否涉及敏感資訊或個人隱私問題，本研究提出了一系列的分析準則如下。



圖二：*AppLeak* 系統架構圖

在使用者 U_i 手機的 Android 應用程式 App_i 相關的 SQLite 資料表 T_i 中，一組資料元素 dt_i 為多個屬性的集合，其中每個屬性 a_j 包含一個標籤 $a_j.l$ 和一個值 $a_j.v$。舉例來說，一位名叫 Bob 的使用者從 Facebook 應用程式[22]相對應的 SQLite 資料表中，萃取出與個人隱私資料有關的資料元素，其中包含 friend_data、user_values、stream_photos、event_notification 等元素。一組資料元素 dt_{Bob}={(user_id, 100003743458), (first_name, Aurora), (last_name, Wang)} 中，其屬性參數 $a_1.l$=user_id、$a_1.v$=100003743458、$a_2.l$=first_name、$a_2.v$=Aurora、$a_3.l$= last_name、$a_3.v$= Wang 代表該使用者的朋友資訊。最後，我們給予每一個屬性一個可以代表其重要性的量化權重 W_{aj}，作為我們對於敏感資訊或個人隱私評估的依據。

如圖二系統架構所示，惡意攻擊意識 AC_i 可能會影響個人認知 IP_i 或資訊遺失 IL_α，而 IP_i 可能會干擾隱私洩漏 PL_β 的判斷。因此，我們先呈現針對資訊遺失與隱私洩漏評估的正確性和完整性，再來提出對於 AC_i、IP_i、IL_α、PL_β 的定義。

對於任何獲取的原始紀錄 r_i 而言，正確性意指原始紀錄 r_i 中的部分資訊，能夠正確地被應用程式 App_i 有關的參考紀錄 re_i 識別，因此，相對於參考紀錄 re_i 的原始紀錄 r_i，其正確性的定義如以下公式：

$$Pr(r_i, re_i) = \frac{\sum_{a_j \in (r_i \cap re_i)} w_{a_j}}{\sum_{a_j \in r_i} w_{a_j}}$$

對於任何獲取的原始紀錄 r_i 而言，完整性意指原始紀錄 r_i 中的部分資訊，能夠實際被使用者 U_i 有關的參考紀錄 re_i 識別，因此，相對於參考紀錄 re_i 的原始紀錄 r_i，其完整性的定義如以下公式：

$$Re(r_i, re_i) = \frac{\sum_{a_j \in (r_i \cap re_i)} w_{a_j}}{\sum_{a_j \in re_i} w_{a_j}}$$

資訊遺失 $\boldsymbol{IL_\alpha}$：針對一個相對於參考紀錄 re_i 的原始紀錄 r_i，給予其正確性與完整性，而資訊遺失 IL_α 定義的公式為 $F_\alpha = \frac{Pr \times Re}{\alpha \times Re + (1-\alpha) \times Pr}$，其中 α 代表一個能權衡 Pr 和 Re 之重要性而預先定義之權重。如果 $F_\alpha > \overline{F_\alpha}$ 的可能性是不容忽視的，則資訊遺失攻擊會因此發動。

惡意攻擊意識 $\boldsymbol{AC_i}$：如果惡意攻擊意識 $AC_i=\{r_1, r_2, \dots, r_m\}$ 的要素被敵人 Eve 取得，則敏感性隱私攻擊就會發動。

個人認知 $\boldsymbol{IP_i}$：一旦一系列特定資料紀錄 $\{r_1, r_2, \dots, r_m\}$ 被敵人 Eve 正確地取得，則視為一個使用者 U_i 的隱私被揭露。

隱私洩漏 $\boldsymbol{PL_\beta}$：給定 $AC_i=\{r_1, r_2, \dots, r_m\}$、$IP_i=\{r_1, r_2, \dots, r_n\}$ 和 Android 應用程式 App_i，如果以下條件成立：$PL_\beta = \beta \times \{F_\alpha\}_{AC_i} + (1-\beta) \times \{F_\alpha\}_{IP_i}$ 以及 $PL_\beta > \overline{PL_\beta}$，其中 $0 \le \beta \le 1$，且 $\overline{PL_\beta}$ 是一個預設值，則會發生與 App_i 有關的隱私洩漏。

隱私風險 $\boldsymbol{PR_\gamma}$：給定 Android 應用程式 App_i、與 App_i 有關的隱私風險 PR_γ 是由 $PR_\gamma = \frac{1}{\sum_{SQLite_i} \frac{\gamma_j}{PL_\beta}}$ 估算出來的，其中 $1 \le j \le k$，而 γ_j 是預先定義的，且 $\sum_{j=1}^{k} \gamma_j = 1$。要注意的是 PL_β 將會針對每個與 App_i 有關的 SQLite 檔案(例如：$SQLite_i$)重新評估。

再來，在 $\overline{PR_{high}}$ 和 $\overline{PR_{medium}}$ 是被預先定義的情況下，我們以 App_i 為基礎訂定以下三個隱私風險程度判定的情況：(1)如果 $PR_\gamma \ge \overline{PR_{high}}$，該 App_i 則處於高度隱私風險；(2)如果 $\overline{PR_{high}} > PR_\gamma \ge \overline{PR_{medium}}$，該 App_i 則處於中度隱私風險；(3)如果 $\overline{PR_{medium}} > PR_\gamma$，該 App_i 則處於低度隱私風險。

肆、系統建置

在系統實作方面，本研究真實數據來進行實驗並觀察隱私洩漏等級，並呈現 *AppLeak* 的實作結果。實驗環境表列如表一，接下來，本研究將描述系統實作程序。

表一：環境描述

行動裝置	ASUS NEXUS 7(Quad-Core 1.2Ghz、RAM 1GB、Android 4.4.2)
開發環境	Eclipse 4.3、JDK1.8.0、Android API 19、ADT 22.6.2

一般來說，Android 中與使用者相關之最普遍且重要的資料都儲存於 SQLite 類型的檔案中，因此，*AppLeak* 必須滲入 Android 系統的底層來獲得基於 SQLite 的檔案。通常，行動應用程式會在 Android 系統中存放數個副檔名為.db 的 SQLite 儲存檔。*AppLeak* 必須先捕捉不同的 SQLite 檔案，並從這些檔案中擷取出敏感資料並加以分析。所有的分析皆可參考預先規範的惡意攻擊意識 AC_i 和個人感知 IP_i，例如：名字、住址、電話號碼、信用卡號碼等，並接著計算出正確度、完整度、資訊洩漏 IL_α、隱私洩漏 PL_β 和隱私風險 PR_γ 等值。如果結果可以有效地辨別出使用者身份，代表存在隱私洩漏的風險。最後，*AppLeak* 會評等 Android 應用程式的隱私風險等級(如高度隱私風險、中度隱私風險或低度隱私風險)。

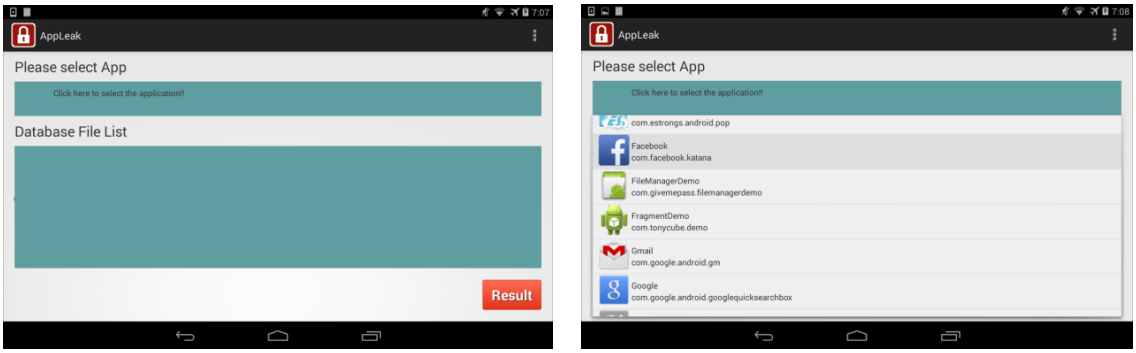
在完成 *AppLeak* 後，我們將利用 *AppLeak* 對兩款熱門的通訊應用程式 Facebook[22] 和 LINE[23]進行隱私風險等級分析。實驗中，本研究使用團隊人員的個人資料內容作為主要分析對象。首先，利用系統來設定 Facebook 和 LINE 的惡意攻擊意識 AC_i 和個人感知 IP_i(如表二)，在與 SQLite 檔案取得的內容進行比對時，本研究指定比對 re_{Sean}={(Name, Sean), (Phone, 0937123456), (Address, Tainan), (Mail, 1000003345678@facebook.com), (Password, XXXXXXXX), (Communication Content, r u free this saturday), (SSN, U123456789), (FB ID, 1000003345678), (Bday, 8/26)}。接下來，我們將呈現 *AppLeak* 的分析結果。

表二：惡意攻擊意識 AC_i 和個人感知 IP_i 的設定

攻擊條件	權重	個別認知	權重
Name	2	**Name**	2
Phone	3	**Phone**	3
Address	3	**Address**	1
Mail	2	**Mail**	1
Password	4	**Password**	5
Communication Content	3	**Communication Content**	4
SSN	5	**SSN**	5
FB / LINE ID	2	**Intimate Friend**	4
B Day	1		

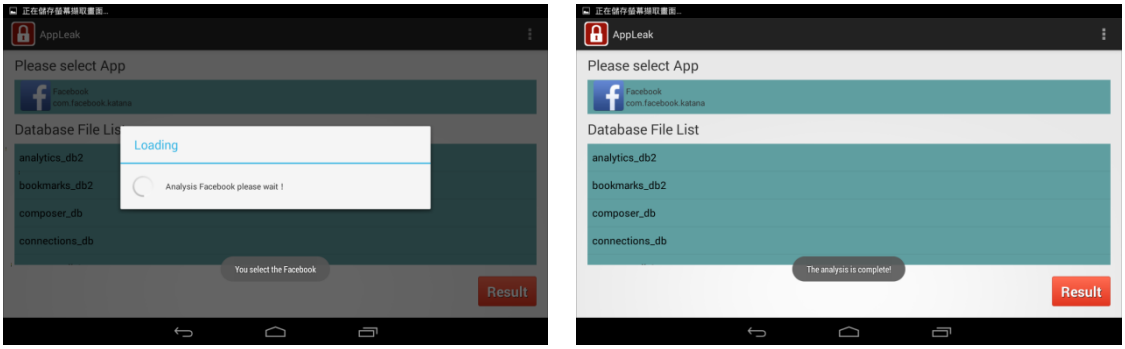
情境模擬(偵測機敏資訊和個人隱私的洩漏)之結果：*AppLeak* 模組捕捉所有在 Facebook 與 LINE 即時通訊軟體下與個人資料相關的 SQLite 檔案內容，並分析這些資料來量化暴露這些資訊的風險。圖三(a)和(b)顯示出使用者可選擇欲分析的 Android 應用程式。圖四(a)和(b)顯示出 *AppLeak* 自動分析 Facebook 所有的 SQLite 檔案，並在列表畫面中顯示分析目標，如此一來使用者就可以知道 Facebook 應用程式下存在多少 SQLite 檔案。接下來，我們選擇 contacts_db2 和 threads_db2 兩個檔案，如圖五(a)和(b)所示。

接著，圖六(a)-(e)提供了細部的分析報告，同時間，圖六(f)亦呈現了 Facebook 資訊洩漏與風險狀態的總覽，並主動通知使用者該應用程式的隱私風險等級。



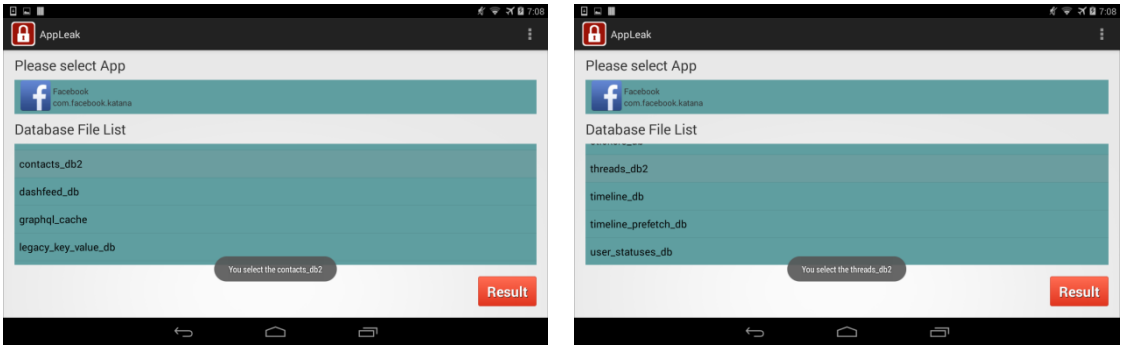
(a)選擇欲進行分析的應用程式　　(b)應用程式列表

圖三：*AppLeak* 模組主畫面



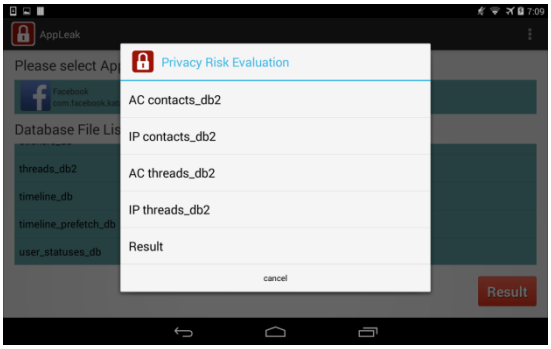
(a)應用程式資料分析中　　(b)完成畫面

圖四：分析 Facebook 的螢幕擷圖



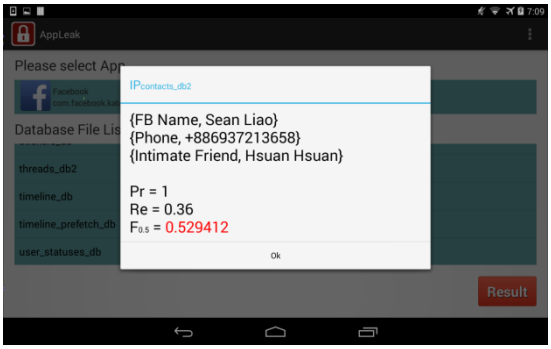
(a)選擇檔案 contacts_db2　　(b)選擇檔案 threads_db2

圖五：選擇 SQLite 檔案時的螢幕擷圖

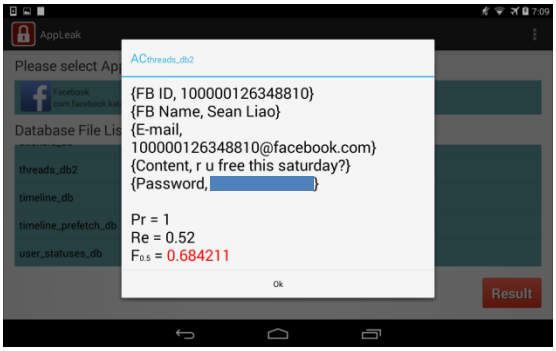


(a)選擇分析結果

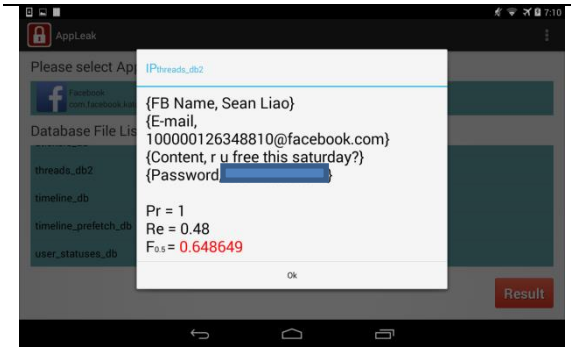
(b) $AC_{contacts\_db2}$ 分析結果



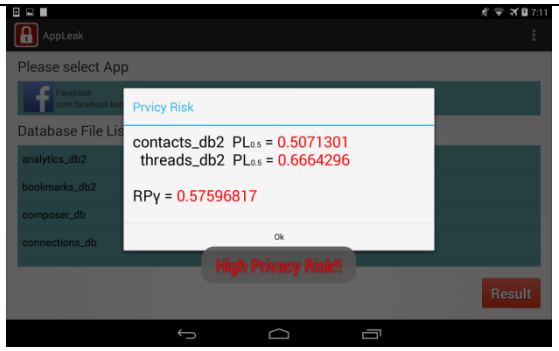
(c) $IP_{contacts\_db2}$ 分析結果



(d) $AC_{threads\_db2}$ 分析結果



(e) $IP_{threads\_db2}$ 分析結果



(f)隱私風險評估結果

圖六：Facebook 分析結果

在實驗中，本研究採用了如表三中的 α、β、γ_1 和 γ_2 共四個權重參數。α 和 β 皆預設為 0.5，因為本研究認為正確度與完整度的重要性是一樣的。然而，γ 的數值會因為不同的 SQLite 來源而影響隱私洩漏，所以 γ 有些許不同。在表四中，我們設定了三個評估條件，以區別隱私風險等級。最後，表五和表六呈現了 *AppLeak* 對 Facebook 和 LINE 的分析結果，結果可見，兩個 Android 應用程式皆存在高度的隱私風險。

表三：權重設定

	α	β	γ_1	γ_2
Facebook 權重	0.5	0.5	0.5	0.5
LINE 權重	0.5	0.5	0.9	0.1

表四：隱私風險評估條件

	高風險 $\overline{PR_{high}}$	中風險 $\overline{PR_{medium}}$	低風險 $\overline{PR_{low}}$
評估條件	$PR_\gamma \geq 0.5$	$0.5 > PR_\gamma \geq 0.25$	$0.25 > PR_\gamma$

表五：Facebook 分析結果

Facebook	$AC_{contacts\_db2}$	$IP_{contacts\_db2}$	$AC_{threads\_db2}$	$IP_{threads\_db2}$
正確度	1	1	1	1
完整度	0.32	0.36	0.52	0.48
資料洩漏	0.484848	0.529412	0.684211	0.648649
γ	γ1 = 0.5		γ2 = 0.5	
隱私洩漏	0.5071301		0.6664296	
隱私風險	0.57596817			

表六：LINE 分析結果

LINE	AC_{never_line}	IP_{never_line}	$AC_{never\_line\_myhome}$	$IP_{never\_line\_myhome}$
正確度	0.846153846	1	0	0
完整度	0.44	0.48	0	0
資訊洩漏	0.578947	0.648649	0	0
γ	γ1 = 0.9		γ2 = 0.1	
隱私洩漏	0.613798		0	
隱私風險	0.6819977777777776			

伍、結論

近來，人們在使用 Android 應用程式時，「提供部分個人隱私資訊以獲取更佳的加值服務品質」此一狀況已逐漸變得無可避免，因此，如何在各種應用服務中或應用程式間的相互作用下，有效地防護個體隱私將成為未來研究社群重要的挑戰之一。在本研究中，我們提出了一個適用於 Android 應用程式的隱私洩漏與風險評估框架 *AppLeak*，框架中採用了惡意攻擊意識與使用者知覺來判定 Android 應用程式的資訊與隱私洩漏，並藉此計算出該應用程式的隱私風險。與既有在 Android 系統上的隱私防護機制相較，*AppLeak* 同時採用客觀隱私認知、主觀隱私概念，並搭配 App 執行程序來進行隱私洩漏的分析與風險評估，此一分析觀點與先前研究並無相關，且可相互整合，其新穎性可見一斑。在系統實作方面，本研究開發了一套風險評估應用程式，並以兩款熱門應用程式(Facebook 和 Line)進行測試，總結出此二應用程式其下的隱私洩漏和風險評估，實驗發現，*AppLeak* 可以有效地針對 Android 應用程式進行隱私風險的評估。

陸、誌謝

本研究接受行政院科技部專題研究計畫(MOST 103-2221-E-259-016-MY2與MOST 103-2221-E-011-090-MY2)之研究經費補助，得以順利完成研究工作，在此致謝。

參考文獻

[1] C. Bi, "Research and Application of SQLite Embedded Database Technology," *WSEAS Transactions on Computers*, vol. 8, no. 1, May 2009, pp. 539 - 543.

[2] D. Biswas, I. Aad, G. P. Perrucci, "Privacy Panel: Usable and Quantifiable Mobile Privacy," in *Proc. of the 8th International Conference on Availability, Reliability and Security*, September 2013, pp. 218 - 223.

[3] D. Chen, X. Han, W. Wang, "Use of SQLite on Embedded System," in *Proc. of the 2010 International Conference on Intelligent Computing and Cognitive Informatics*, June 2010, pp. 210 - 213.

[4] W. Enck, P. Gilbert, B. G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proc. of the 9th USENIX conference on Operating systems design and implementation*, no. 1-6, June 2010, pp. 1-6.

[5] M. Frank, B. Dong, A. Porter-Felt, and D. Song. "Mining Permission Request Patterns from Android and Facebook Applications," in *Proc. of the 2012 IEEE International Conference on Data Mining*, December 2012, pp. 870-875.

[6] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale," in *Proc. of the 5th international conference on Trust and Trustworthy Computing*, 2012, pp. 291-307.

[7] M. Grace, Y. Zhou, Q. Zhang, S. Zou, X. Jiang, "RiskRanker: scalable and accurate zero-day android malware detection," in *Proc. of the 10th international conference on Mobile systems, applications, and services*, June 2012, pp.281-294.

[8] H. Kuzuno, S. Tonami., "Signature Generation for Sensitive Information Leakage in Android Applications" in *Proc. of the 29th IEEE International Conference on Data Engineering Workshops*, April 2013, pp.112 - 119.

[9] M. Landman, "Managing Smart Phone Security Risks," in *Proc. of the 2010 Information Security Curriculum Development Conference*, Oct. 2010, pp.145-155.

[10] J. Lv, S. Xu, Y. Li, "Application Research of Embedded Database SQLite," in *Proc. of the 2009 International Forum on Information Technology and Applications*, May 2009, pp. 539 - 543.

[11] C. D. Manning, Pr. Raghavan, and H. Schtze, *Introduction to information retrieval*, Cambridge University Press, NY, USA, 2008.

[12] N. A. Mutawa, I. Baggili, A. Marrington, "Forensic analysis of social networking applications on mobile devices," in *Proc. of the 12th Annual Digital Forensics Research Conference*, vol. 9, August 2012, pp. 24-33.

[13] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: versatile protection for smartphones," in *Proc. of the 26th Annual Computer Security Applications Conference*, December 2010, pp. 347-356.

[14] S. Rosen, Z. Qian, and Z. M. Mao, "AppProfiler: a flexible method of exposing privacy-related behavior in android applications to end users," in *Proc. of the 3rd ACM conference on Data and application security and privacy*, February 2013, pp. 221-232.

[15] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, 2001, pp. 1010 - 1027.

[16] P. Samarati and L. Sweeney, "Generalizing data to provide anonymity when disclosing information," in *Proc. of the 7th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 1998, pp. 188.

[17] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness, and Knowledge-Base Systems*, vol. 10, no. 5, October 2002, pp. 571 - 588.

[18] N. Ulltveit-Moe, T. Gjøsæter, S. M. Assev, G. M. Køien and V. Oleshchuk, "Privacy Handling for Critical Information Infrastructures," in *Proc. of the IEEE International Conference on Industrial Informatics*, July 2013, pp. 688 - 694.

[19] S. E. Whang, and H. Garcia-Molina, "A model for quantifying information leakage," *Lecture Notes in Computer Science*, Vol. 7482, 2012, pp. 25-44.

[20] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang, "AppIntent: analyzing sensitive data transmission in android for privacy leakage detection,". in *Proc. of the 2013 ACM SIGSAC conference on Computer & communications security*, November 2013, pp. 1043-1054.

[21] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, W. Zou, "SmartDroid: an automatic system for revealing UI-based trigger conditions in android applications," in *Proc. of the 2nd ACM workshop on Security and privacy in smartphones and mobile devices*, October 2012, pp. 93-104.

[22] Facebook Corporation, https://www.facebook.com/facebook

[23] Naver corporation, http://www.navercorp.com/en/index.nhn