以 Andorid 實驗案例探討 OWASP 行動裝置應用程式之十大威脅

許振銘, 許登凱

健行科技大學資訊工程系所

chenming@uch.edu.tw, m10113021@uch.edu.tw

摘要

隨著智慧型行動裝置應用程式與網路服務的快速發展，包含休閒娛樂、社交網路、金融購物等豐富且方便之功能，因此行動網路服務已成為人們生活中的一部分。然而，應用程式開發者與使用者因新技術、新架構、新應用之因素，缺乏相關資訊安全知識，致使遭受行動應用所帶來的嚴重威脅，使得行動網路與服務成為目前最大之資訊安全問題。而上百萬爆炸性成長的惡意程式中，有九成以上的目標都是全球擁有高達八成市佔率之 Android 行動裝置。孫子兵法云:「知彼知己，百戰不殆」，要能有效提升防禦駭客攻擊的能力，必須以駭客思維從實務案例中學習。因此本論文主要以 OWASP Mobile Top 10 所提出之十大行動裝置應用程式風險為中心，並結合多種資訊安全弱點實驗平台(Hands-on Labs)，使用篩選過實驗案例來實務探討與說明每個風險，希望藉此能強化程式開發者的資訊安全程式設計能力外，亦能有效降低使用者遭受惡意程式威脅之風險。除此之外，我們亦詳細介紹了各種行動裝置之資訊安全實驗案例平台外，也包含相關運作環境與工具來進行實驗案例之實作演練。

關鍵詞：Android、資訊安全、行動裝置、行動裝置應用、資訊安全實驗、弱點系統、OWASP Mobile Top 10、惡意程式、行動網路安全、網路服務安全、安全性程式設計。

壹、前言

隨著智慧型行動裝置網路環境、行動裝置應用程式(Mobile Applications, 簡稱 APP)、與行動網路服務爆炸性的成長，引領人們進入了行動化生活的時代，漸漸地取代了傳統手機與電腦，至今已成為網路應用的前端設備主流市場，人們可利用智慧型行動裝置來進行休閒娛樂、金融服務、線上購物、資訊管理、與社交網路等。行動網路使用者與 App 開發者的資訊安全知識並沒有跟上行動網路的快速發展，致使行動裝置與服務已成為惡意駭客的主要攻擊目標，因此行動資訊安全已成為目前最重要的研究議題[1][2][3][4]。

目前行動裝置的使用者數量已遠比個人電腦還多，其普及速率是個人電腦的十倍，也是網路化的三倍，其中佔半數以上為 Android 系統，全球 Android 設備已超過 9 億台，其 APPs 數量也突破百萬大關，每年超過 700 億 APPs 的下載量，且在 2012 年至 2013 年

之間有近千萬 Android 使用者遭受惡意程式攻擊。行動裝置之惡意程式的成長與演進速率比傳統 Windows PC 快四倍以上，僅 3 年時間就達到 Windows 系統 14 年的規模，其成長率也從 2011 的 155%翻倍至 2012 年的 614%，並有超過 500 個包含惡意程式之第三方 app stores。至今行動裝置惡意程式已突破百萬，而其中有 9 成以上之攻擊目標皆為 Android 系統[5][6][7][8]。

行動裝置之惡意程式主要有四大威脅類型，包含提供後門讓未授權使用者可以存取設備(Backdoor)、使用簡訊進行付費購買或濫用等惡意行為(SMS)、偷取資料或收集帳號密碼(SPY)、下載並執行惡意程式(Download) [9]，其中有 73%皆為 SMS 惡意程式[8]。例如 Fakeinst 與 Opfack 都是偽裝成合法的程式來傳遞簡訊到高付費號碼或訂閱昂貴服務。另外，GinMaster 為目前風險極高且經常被使用於嵌入於合法程式中的木馬程式[4]。

使用者與智慧型行動裝置形影不離，但相關資訊安全素養顯然非常不足，因為有大量使用者沒有採取基本預防措施使得個資暴露於風險中，同時使用於工作與娛樂導致企業面臨全新的安全風險，且認為雲端儲存是安全的，也會使用公開且不安全的無線網路。由此可見，大部分使用者完全或幾乎沒有意識到使用行動裝置所可能帶來的威脅。

工欲善其事，必先利其器，因此欲學習與實踐行動裝置之資訊安全實務，並非僅吸收相關資安知識就能有效學習與防禦，而是需透過資訊安全案例才能進行實務演練[10][11][12][13]。美國五角大廈訓練工作人員侵入自己的國防部電腦網路，目的在協助工作人員從真正的駭客經驗中學習如何保護自己的電腦系統。但顯然的我們不能因為要學習而明目張膽地入侵他人的行動裝置。而傳統 PC 與伺服器在資訊安全實驗案例(hands-on labs)相關研究與資源非常充足，國內發展了很多攻防訓練之研究實驗平台提供案例實務之培訓外 [14] [15]，也有不可計數的開放式駭客競技學習實驗平台[16][17][18]。

反觀行動裝置上之資訊安全實驗平台，雖然 EC-Council 於 2013 年於全球最知名的道德駭客國際認證 CEHv8 課程中，特別加入了全新的 Hacking Mobile Platforms 教學單元外；全球知名的開放 Web 軟體安全計畫社群(Open Web Application Security Project, OWASP)亦於 2012 年率先提出了行動裝置十大風險計畫(OWASP Mobile Top 10 Risk, 簡稱 OMT10) [19]，但依然缺少行動裝置之資訊安全實驗案例提供實務學習。

為了能夠提供行動裝置之資訊安全實驗案例進行實務學習，Security Compass Labs 率先於 2011 年以開放式源碼釋出了針對行動裝置應用程式之資訊安全實驗案例訓練平台(Exploit-Me - Exploit Mobile Android Labs, Exploit-Me)[20]，同年八月 Honeynet Project 舉辦並釋出了行動裝置惡意程式分析之挑戰活動(Honeynet Project Challenges - Mobile Malware)[21]，2012 年同時有 Jack Mannino 釋出所開發的 OWASP GoatDroid Project(FourGoats 與 Herd Financial)[22]、McAfee 的 Hacme Bank Android(簡稱 HacmeBank)[23]、以及 Paladion's AppSec tools for Mobile Enthusiasts (簡稱 InsecureBank)的行動裝置資訊安全實驗平台[24]。上述實驗平台皆為主從式架構，除 OWASP Fourgoats 為位置社交網路弱點應用程式外，其他皆以行動銀行應用為案例所設計之具有弱點的

APP，用以讓 APP 開發者學習如何開發更安全的應用程式來降低不安全的應用程式釋出外，亦可增強一般使用者在應用上的資安常識以減少遭受惡意程式之威脅與攻擊。

雖然 OMT10 從大量資安事件中歸納出十大 APP 之風險，但目前之內容尚屬於雛型階段，並無提供完整且詳細之說明與案例，因此無論開發者或使用者皆難以從中理解或實際體驗 OMT10 其提出之十大風險。雖然近年來有少數相關之資訊安全實驗平台可提供實務演練，但皆不以 OMT10 所提出之十大風險來進行設計，例如 OWASP GoatDroid 本身也僅提供部分風險之實驗案例，並沒有涵蓋全部之風險案例。

有據以此，本論文主要以 OMT 10 所提出之 APP 十大風險，我們結合各種不同資訊安全實驗平台所提供之案例，針對每一項風險進行實務案例探討，藉以讓開發者與使用者皆能從實驗案例探討中了解其 APP 開發與使用之風險。首先介紹 Android 資訊安全實驗滲透環境，再重新重點釋義 OWASP 十大 APP 風險，接著詳細介紹各種 APP 之資訊安全實驗平台，在以篩選的資訊安全實驗案例來探討每一個 OWASP APP 風險，最後結論。

貳、Android 資訊安全實驗案例環境

Android 應用程式運作機制

Android 是由 Google 以 Linux 系統為基礎所開發出應用於行動裝置(例如手機與平板電腦)之多行程系統，Java 為 Android 應用程式主要之開發語言，Android 應用程式是透過 Dalvik 虛擬機器(Dalvik virtual machine, DVK)來執行，而非使用 Java 虛擬機器(Java virtual machine, JVM)，因此 Java 程式碼編譯成.class 檔案後還需要經由 Android SDK[25]的 dx 工具轉換成可以在 DVK 執行的相同代碼.dex(Dalvik EXecutable)與.odex(Optimized Dalvik eXexutable)。Android 的應用程式模型(Application model)以 apk(Android application package file)之壓縮安裝檔形式存在，當應用程式啟動時會建立 Task 堆疊空間，一個 Task 包含一或多個使用者互動的物件視窗 Activities，兩個 Activities 要進行跳轉或傳遞參數則是透過 Intent 來進行行程間通訊(IPC)，而 Activity 透過 View 類別繪製介面(UI)與處理事件產生互動式使用者介面，UI 則可另透過 XML layout (Layout.xml 設定檔)進行介面之排版。系統和應用程式之間的安全性除了透過 Linux 程序級別來強制實現讓每個應用程式分配到特定使用者與群組 ID 外(預設 root 帳戶停用)，亦以 Permission 機制限制應用程式所能允許的操作許可權限(可於 AndroidManifest.xml 設定檔中設定需求之權限，同時透過沙箱對應用程式間進行隔離來保護不被其他應用程式非法存取內部資訊。

Android SDK 應用程式開發環境與滲透工具

Android SDK(Software Develop Kit)是提供 APP 的開發環境，其中的 ADT (Android Developer Tool) Bundle 包含了 Android SDK Tools、整合型開發環境 Eclipse+ADT plugin (Java ADT)、Android Platform-tools、與 Android emulator(模擬器)。藉由 SDK manager 可

使用所有的 Android 系統版本,在透過 AVD(Android Virtual Device) manager 建立 Android 模擬器,可選擇需要的 Android 版本、RAM 大小、行動裝置型號、SD card 空間、與 Camera 等功能。雖然 AVD manager 可直接開啟 Android 模擬器，但還可以透過 emulator 程序之參數來達到更多的功能，如 emulator [-http-porxy]可直接設定 proxy (後續以方括號[]代表參數)、[-tcpdump]可錄製封包、[-logcat]啟用 log 功能、與[-shell]進入 root terminal 等。

為了方便管理 Android 模擬器還提供了 ADB (Android Debug Bridge)工具，如 adb [shell]可以進入 Android shell 模式進行指令之操控，參數[procrank]可以觀察執行中程序、參數[strace]可用來除錯與追蹤特定行程之系統與信號呼叫(可特別運用於無法使用 Proxy 偵錯之 APP)，參數[logcat]可取出系統日誌，參數[pull]可將行動裝置(模擬器)中之檔案傳至本機，參數[push]可將本機檔案傳至行動裝置(模擬器)等功能。然而 Android shell 所提供的指令與功能有限，但藉由 busybox[26]則可提供更多元的指令功能於 Android shell 模式中執行，能提升對 Andorid 系統的操控性。另外亦可使用 Android 提供的 sqlite3 指令搭配 SQL 語法操作裝置中的資料庫檔案，但我們也可透過 Windows 平台的 SQLite Database Browser 工具[27]，以方便的操作介面存取行動裝置資料庫。

Android 系統模擬

測試 APP 無需使用實體智慧型行動裝置，可透過模擬器(emulator)來模擬 Android 系統並將 APP 執行於模擬環境中。目前模擬工具有 Android SDK (Software Develop Kit)、Android NDK[28] (Native Development Kit)、與 Android-x86[29]等。Android SDK 是以 JAVA 為基礎的整合型 APP 開發環境，而 Android NDK 則是另一個以 C / C++為基礎的整合型 APP 開發環境，兩者皆支援所有 Android 系統版本。Android-x86 則是提供 x86 架構的 Android ISO 映像檔，可獨立安裝運行於虛擬軟體上(如 VirtualBox[30])，相對於 Android SDK 模擬環境，Android-x86 可模擬更真實的環境，支援 USB 型之 Wi-Fi 網卡、GPS 接收器、藍芽與 SIM 讀卡機等。縱使 Android SDK 僅提供兩個模擬器之間的電話與簡訊功能與固定式 GPS 功能，但已可滿足本論文資訊安全實驗案例演練之目的。

Android 應用程式安全性分析

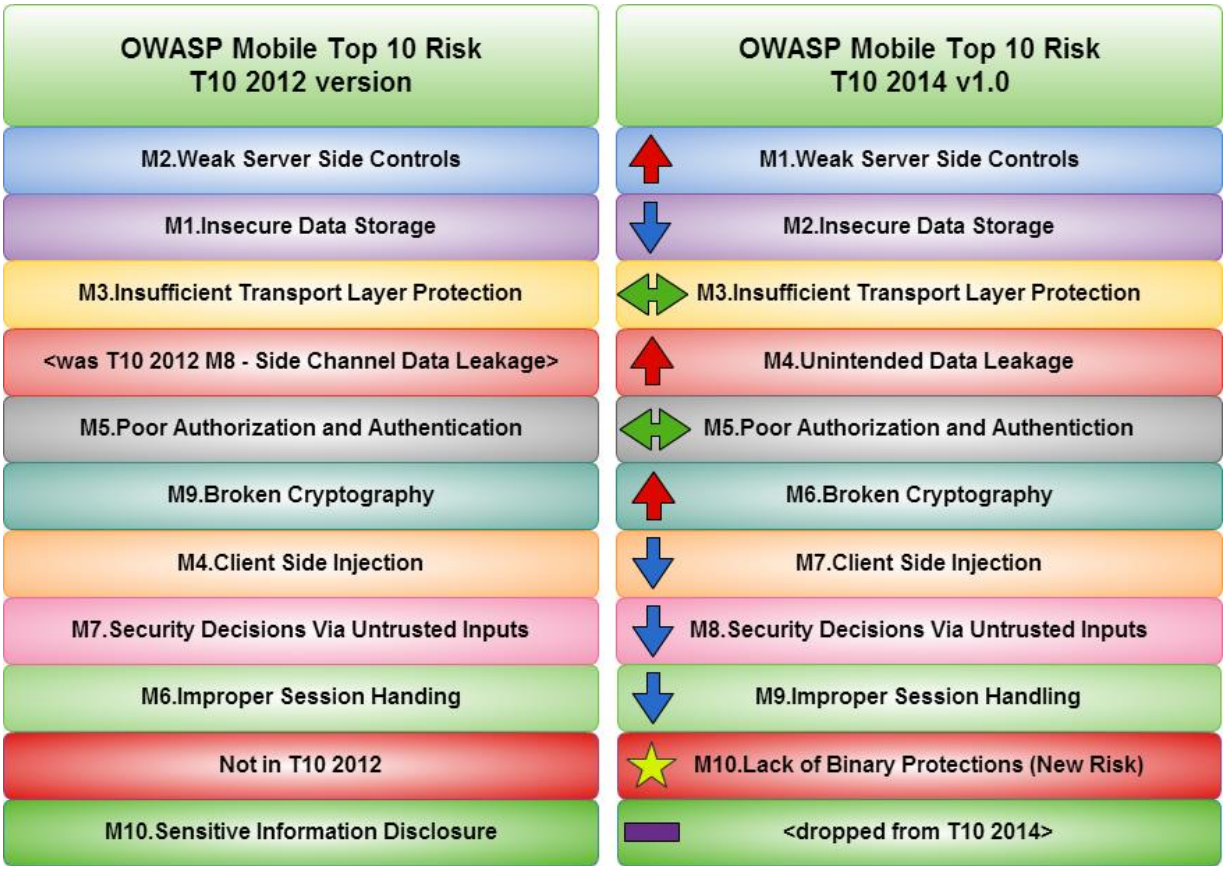
APP 分析方式大致可分為「靜態分析」與「動態分析」,所謂靜態分析指的是在不執行應用程式情況下，單純就程式碼內容進行分析，僅能分析到程式碼靜態的特性，可透過 apktool[31]對 apk 檔進行反組譯取得應用程式的.smali 檔案(Dalvik bytecode)與相關設定等檔案，例如有包含應用程式資訊與需求權限之 AndroidManifest.xml 等。從應用程式的 JAVA 原始碼觀察才能有效了解其設計邏輯,然透過反組譯的.smali 格式檔是難以從中分析的。首先解壓縮 apk 檔取出.dex 檔,在使用 dex2jar[32]對.dex 檔進行反編譯成.jar 檔，接著再透過 JD-GUI[33]將.jar 檔反編譯成接近 apk 之 JAVA 原始碼進行分析與觀察找出關鍵資訊，最後依據找到的關鍵資訊針對.smali 檔進行竄改，再透過 apktool 將修改過後的.smali 檔案組譯回 apk 安裝檔。相反的，動態分析就是在分析應用程式運行時的狀態，

可實現靜態分析無法達成的功能，例如記憶體洩漏資訊(Memory leak)或是非法記憶體存取。藉由 Java ADT 的 DDMS(Dalvik Debug Monitor Server) [34]的 dump hprof 功能即可傾印出 Android APP 執行期間的記憶體資訊，然而因為 APP 運作於 DVK 上，所以 HPROF dumps 並非正確的 Java 格式，所以需再經由 Android SDK 提供的 hprof-conv 工具將 Android HPROF dumps 進行轉換，最後載入至 Eclipse MAT(Memory Analyzer Tool)[35]進行分析，即可觀察 APP 的記憶體相關資訊。

Android 應用程式網路行為分析

在 APP 之網路分析方面，經常以建置本地代理伺服器 Proxy 來攔截應用程式的網路流量封包進行滲透測試，常見的整合型 Web 滲透軟體有 Burp Suite[36]、OWASP Zed Attack Proxy(ZAP) [37]等，Web 除錯工具有 WebScarab[38]與 Charles[39]等。上述軟體皆提供 Proxy 功能，可針對特定 APP 或行動裝置進行網路封包攔截、竄改、偵錯與測試。

參、OWASP 十大行動裝置風險



圖一、2012 與 2014 年之 OWASP 行動裝置十大風險比較對照圖

行動裝置的風險涵蓋的範圍包含網路層次(WiFi 或 GSM 漏洞)、硬體層次(例如在韌體中透過記憶體錯誤漏洞取得管理者權限)、作業系統核心碼或供應商支援系統碼(例如越獄(Jailbrake)、應用程式與雲端服務。在應用程式面，Web 軟體安全計畫(Open Web Application Security Project, 簡稱 OWASP) [40]於 2011 年透過行動裝置威脅模型研究方法(Mobile Threat Model)發表了 OWASP 行動裝置安全之十大威脅(OWASP Mobile Security Project - Top Ten Mobile Risks)，依其將遭受的威脅分成六大類：分別為詐欺(Spoofing)、否認(Repudiation)、阻斷服務(Denial of Service)、竄改(Tampering)、 資訊洩漏(Information Disclosure)、權限提升 (Elevation of Privilege)，並進一步依據上述分類之風險衝擊性定義十大風險，目前為 2014 v1.0 版本，圖一顯示 2012 年與 2014 十大風險之對照表與威脅等級變化，上箭頭、下箭頭、雙箭頭、橫線、與星號於 2014 年之標示分別代表威脅等級增加、下降、沒有改變、刪除、與新增。依威脅等級順序重點簡述如下：

2014-M1: 伺服端缺乏完善的控管(Weak Server Side Controls)

智慧型行動裝置之應用程式多依賴傳統 Web 伺服器提供驗證與資訊之服務，因此 APP 之網路服務依然面臨著現今 PC 伺服器所存在之威脅，例如在 OWASP Web Top 10 與 Cloud Top 10 亦將此列為高風險之一。其問題在於 APP 之伺服端缺乏完善之資安控管，例如伺服端授權與驗證不嚴謹或邏輯層設計不夠周延，致使 APP 或使用者可傳送未受信任之資料至 Web 伺服器，或更甚者直接透過 API 或網站服務介面直接呼叫來進行惡意行為。伺服端資安管控不當所造成的資安問題因素有三項：(1) APP 程式開發框架是以容易開發所設計的，並沒有將資訊安全列為優先考量；(2) 開發商對於新開發環境與新行動應用缺乏相關資訊安全知識外，亦貪圖方便與節省成本而沒有經過完善之資安規劃與實作測試下魯莽上市 APP；(3) APP 都是透過 PC 平台進行跨平台開發與編譯，致使無法測試與考量開發之 APP 安裝於實體行動裝置與系統中可能存在的資安漏洞。因此， APP 之伺服端進行完善的資訊安全考量，例如參考 OWASP Web Top 10 與 OWASP Cloud Top 10 所提供之伺服端資安控管建議，並將之實作與測試才能有效降低此資安威脅。

2014-M2: 儲存的資料無安全防護(Insecure Data Storage)

由於行動設備之功能與 APP 的應用越來越多元，致使越多的個人機敏資訊儲存於行動裝置中，一但行動設備遺失或存在惡意程式即可能造成資訊被盜取之風險。雖然預設行動裝置之系統除使用沙箱機制隔離應用程式進行資料保護外，亦封鎖可存取內建儲存裝置之檔案系統 root 權限，然一但當行動裝置有經過 Rooting 或越獄，或存放於抽取式 SD 儲存卡中，則可透過電腦並搭配相關工具直接存取行動裝置檔案系統來取得所有 APP 目錄下的資訊。行動裝置之檔案系統通常包含著非常多的個人機敏資訊，常見儲存於 SQLite 資料庫、系統紀錄 Log、紀錄網站連線狀態與資訊之 Cookie、抽取式 SD 儲存卡、雲端同步、XML 設定檔、與 APP 整體資訊設定檔 Manifest，內容之資訊可能包含個人帳號密碼、認證權杖(Authentication tokens)、Cookies、GPS 位置資訊、UDID/EMEI、設

備名稱、網路連線名稱、個人相關資訊等。為保護儲存於行動裝置之資訊，開發者必須要能意識到其開發之 APP 所儲存資訊可能會有被盜取之風險，非必要不要儲存個人機敏資訊於行動裝置中；如果因其應用而需要儲存，除避免儲存於 SD 卡外，也要將之設定為私有資訊狀態(亦即不可啟用 MODE_WORLD_READABLE 標記)，同時搭配主密碼(Master password)與進階加密標準 AES 128 來針對機敏資訊進行加密。

2014-M3: 傳輸層無足夠之防護(Insufficient Transport Layer Protection)

大部分的 APP 應用皆為主從架構，因此皆會透過使用者端應用程式跟伺服端進行通訊傳輸，WiFi 與 3G 網路為目前常用之網路通道，如開發商在應用程式開發時沒有完善的安全傳輸考量與實作，則攻擊者即可從傳輸資料的過程中竊取或竄改封包內容來取得使用者機敏資訊或進行中間人攻擊(Man-in-The-Middle, MiTM)。盜取傳輸資訊的方法包含使用者在區域網路中的 WiFi 已被監控攔截、資訊在傳輸載體(如基地台、路由器、代理伺服器等)中被竊取、已被植入惡意軟體於行動裝置中。目前大部分行動裝置上的 APP 不是以明文方式在網路上傳輸資料，不然就是僅針對認證才使用 SSL/TLS 進行傳輸資料加密。就算有全面性的 SSL/TLS 加密傳輸，依然存在設定與管理不當之問題，因此亦容易遭受釣魚攻擊(Phishing)或中間人攻擊。因此最好的傳輸層防護方法是應用程式在釋出前先透過 Proxy 來確認是否所有的連線傳輸皆有加密外，其所使用之 SSL 憑證也必須詳加檢查，例如有效日期、是否為自我簽章(Self-signed)、以及加密強度是否足夠。以 Android 為例，建議在開發週期移除如下設定，不允許接受應用程式憑證：

(1) Org.Apache.Http.Conn.SSL.AllowAllHostnameVerifier

(2) SSLSocketFactory ALLOW_ALL_HOSTNAME_VERIFIE

2014-M4: 快取或暫存資料被竊取(Unintended Data Leakage)

快取或暫存資料被竊取的風險又稱為側通道資料洩漏(Side-channel data leakage)，為 2014-M2 不安全資料儲存的分支，但其著重於作業系統、開發框架、編譯環境、與硬體所造成的快取資料(Cached data)洩漏。快取之資料包含 URL、鍵盤的輸入、拷貝/貼上、系統紀錄、HTML5 資料儲存、瀏覽器 Cookie、執行於背景的應用程式、以及傳送給第三方的分析資料等。然目前這些快取的機制不是沒有文件說明，就是未公開，因此應用程式開發者無從了解，也當然無法實作相關機制去管理與保護快取資料。雖然最佳的防護機制即是必須了解所有軟硬體與開發環境之快取機制，才能有效降低快取或暫存資訊被竊取之風險，但由於快取機制大部分皆為黑箱作業，開發者與一般使用者皆難以掌控，但透過可信任之第三方快取資料移除與保護之專業軟體進行防護才是最佳之解決方案。

2014-M5: 不嚴謹的授權機制與身份認證(Poor Authorization and Authentication)

使用端本機認證將可在已越獄的設備上，於程式執行期間使用除錯工具進行竄改與操作程式之二進位程序來通過驗證。因此，應用程式開發者除應由伺服端全面進行認證

工作外，尚需考量幾個防護機制：(1)不要實作 Remember Me 功能來方便使用者登入，即持續性認證(Persistent authentication)，此功能所紀錄之帳密會有被盜取之風險；(2)如用戶端有儲存認證資訊必要性，則必須將之加密；(3)不要使用任何可假造(Spoof-able)的資訊進行認證，例如設備識別號或 GPS 位置資訊，這些資訊都可以被假照來欺騙認證系統。(4)多善用以特定設備為主之認證權杖 (Device-specific authentication token)，如此即可在設備遺失或被盜取後透過網站應用程式進行註銷，例如 Facebook 即可透過網站內的帳號保安的功能中註銷特定裝置認證權杖。如此才能強化認證與授權的資訊安全。

2014-M6: 加密演算法不嚴謹或被破解(Broken Cryptography)

無論是 APP 資訊儲存或通訊傳輸，無使用或使用容易被破解的加密演算法時，將增加資料被盜取並破密成明文的風險。例如 APP 使用較弱之密碼並以 MD5 演算法進行加密，則攻擊者可透過預計算雜湊之彩虹表資料庫與相關工具[41]進行離線密碼破解 MD5 加密之密碼。該風險產生之主要因素有: (1) 開發者使用了較不安全的加密演算法，例如 RC2, MD4, MD5, 與 SHA1；(2) 金鑰管理不當，例如將金鑰與加密資料放置在相同目錄中，或使用了攻擊者也擁有的相同金鑰；(3) 錯誤使用內建之加密函式；(4) 將金鑰以 Hard-coded 方式存放於 APP 程式碼中。因此開發者應該避免上述的不當管理與錯誤使用，建議使用資安社群(Security community)所接受之強健且較新演算法進行加密管理。

2014-M7: 用戶端注入攻擊變造(Client Side Injection)

只要用戶端應用程式沒有針對輸入的資料進行完整之檢測，則可透過外部使用者、內部使用者、或程式本身依據目標直譯器的語法來傳送未信任之惡意文字資料至系統中來達到用戶端注入攻擊之目的。攻擊之目標包含在用戶端欄位輸入 SQL 語法針對 SQLite 進行 SQL Injection 攻擊、針對瀏覽器以 JavaScript 如 XSS 進行跨網站腳本攻擊來取得認證之 cookie 進行連線劫持或目錄走訪攻擊(Local File Inclusion)取得 APP 內部資訊。因此開發者必須針對應用程式端的輸入資料進行所有層面之檢測與確認才能有效降低此類之攻擊風險。以 Android 系統為例，使用透過參數來給值(Parameterized queries)進行資料庫連結與存取則可有效防禦 SQL Injection 攻擊、確認應用程式在 WebViews 中不啟用 JavaScript 與 Plugin 之功能支援外，並同時設定禁止檔案系統存取設定以避免 XSS 與目錄走訪攻擊 “webview.getSettings().setAllowFileAccess(false);”、對 APP 之 Activities 所傳遞之資料進行確認並使用 Intent Filter 過濾來防範 Intent Injection 與漏洞掃描 Fuzzing。

2014-M8:針對不受信任的輸入之不當資安處置(Security Decisions Via Untrusted Inputs)

此風險主要是 APP 允許接受透過所有來源管道所提供的資料，如果沒有進行適當防護，Android 系統之 APP 可以透過 Intent 以 IPC 機制叫用並傳遞參數給其他 APP 行使惡意行為。因此 APP 在 IPC 的防護機制實作上建議僅允許受信任的應用程式、IPC 機敏相關之行為應再次透過使用者確認後才進行、對於所有 IPC 的輸入資料都應進行完善之資

訊安全確認、不以 IPC 機制傳送機敏資訊等。在 Android 平台中，亦可透過權限與 Intents 型態(Intents types)來進行控管哪些應用程式可以叫用 APP 的元件(Components)。

2014-M9: 不當的連線會話管理 (Improper Session Handling)

該風險類似 M5 之弱驗證機制，主要因為伺服端跟用戶端的連線會話(Session)管理不夠嚴謹，例如伺服端沒有機制處理無效之連線會話、將連線會話之驗證交由用戶端程式處理、沒有進行連線會話之有效時間控管、沒有考慮連線會話時使用者身分之轉換、使用不安全的機制建立認證權杖等因素，致使惡意人士有機會可以透過連線劫持(Session hijacking)以存取實體行動裝置或擷取網路 HTTP/S 與 cookie 資料等封包資訊方式劫持雙方連線，導致受害者帳戶失去存取權、帳戶被盜用、資料遺失、機敏資訊被竊取等。因此由伺服端統一控管連線會話，並同時考慮時間有效性(例如 15 分鐘即為高安全性機制)、在同一個連線會話下如使用者身分轉換則需進行重認證(例如匿名使用者轉換至已登入的使用者帳號)、以及使用高安全性並符合企業標準的方法來建立認證權杖。

2014-M10: 應用程式缺乏二進位防護(Lack of Binary Protections)

2014-M10 為 2014 年新增之風險[42]，以 Android 系統應用程式來說，主要之威脅在於 APP 編譯之二進位沒有進行適當的保護，致使有心人士可以透過相關工具以逆向工程技術針對程式進行行為分析與修改。未受保護的二進位碼為目前 APP 最大的問題，因為可被運用來尋找弱點、置入惡意功能、破壞程式驗證機制、與竊取撰寫在程式碼中的機敏資訊如維護用之後門帳密或金鑰等。因此，APP 之設計必須能避免被使用靜態或動態之分析技術進行逆向工程分析，而 Arxan[43]即為用來防護應用程式二進位碼之技術。

肆、Android 資訊安全實驗案例(Hands-on labs)發展現況

本章節主要重點介紹 Android 各種實驗平台之架構與特色案例、Android 資訊安全實驗案例整合環境、以及行動裝置惡意程式分析挑戰平台，所有案例詳列於表一，而表二所有實驗案例與對應之 OWASP Mobile Top 10 風險的分類。

APP 資訊安全實驗平台

OWASP 在 2011 年於美國發表關於行動裝置的十大弱點風險(OMT10)後，緊接著開發並以開放式原始碼計畫(GPLv3 license)釋出 OWASP GoatDroid Project。GoatDroid 是以 OMT10 所歸納的大十大風險為原則來設計的兩個具有多種弱點的 APPs，包含以位置基礎的社交網路(Location-based Social Network) FourGoats 與行動銀行 Herd Financial，以上兩個應用程式一共存在 13 項弱點設計，但由於官方並沒有提供相關實驗案例之說明文件，因此我們透過分析原始碼來了解如何利用其案例漏洞並進行利用。由於 GoatDroid 設計之弱點案例僅涵蓋部分 OMT10 之風險，因此我們僅採用 FourGoats 中的 SQLite 資

料庫儲存未加密之身份認證資訊漏洞、APP 行程間通訊不夠嚴謹導致第三方 APP 盜用相關權限漏洞、與儲存於裝置中之機敏資訊權限為公開讀取(MODE_WORLD_READABLE)等漏洞案例。因此我們將採用與篩選其他資訊安全實驗平台來做為其他風險之案例。

表一：資訊安全實驗平台案例列表

Exploit-Me Mobile Labs	Hacme Bank Android Labs	OWASP FourGoats(F)/ HerdFinancial(H)	Paladion Labs InsecureBank	Honeynet Challenge 9 - Mobile Malware
E1-Secure Connections E2-Parameter Manipulation E3-Insecure file storage E4-Secure logging E5-Basic Encryption E6-Advanced Encryption E7-Memory protection E8-Client-side Password complexity	B1-SQL injection B2-Unauthorized Read B3-Unauthorized Write B4-Unauthorized Execute B5-Application Logic Bypass B6-Insecure Data Storage B7-Excessive Application Permissions B8-Hidden Credentials B9-Proxying HTTPS Traffic When Certificate Validation Is Enabled B10-Defeating SSL Certification Validation B11-Defeating Authentication	F1-Client-Side Injection F2-Server-Side Authorization Issues F3-Side Channel Information Leakage F4-Insecure Data Storage F5-Privacy Concerns F6-Insufficient Transport Layer Protection F7-Insecure IPC H1-Poor Authentication H2-Business Logic Flaws H3-Server-Side Authorization Issues H4-Insecure IPC H5-Insecure Data Storage H6-Insufficient Transport Layer Protection	P1-Information Sniffing due to Unencrypted Transport medium P2-Sensitive information disclosure via Property Files P3-Sensitive information disclosure via SD card storage P4-Sensitive information disclosure via SQLite DB P5-Sensitive information disclosure via Device and Application Logs P6-Sensitive information disclosure via Side Channel Leakage P7-Malicious Activity via parameter Manipulation P8-Malicious Activity via Client-side XSS P9-Malicious Activity due to insecure WebView implementation P10-Sensitive information leakage due to hardcoded secrets P11-Sensitive information leakage due to weak encryption algorithm P12-Malicious Activity via Backdoor P13-Malicious Activity via Reverse Engineering	N1- Write an executive summary of this incident N2-Provide the phone brand, model, OS name and version N3- Extract any suspicious application (if any). Detail your extraction method. Please provide name and SHA1 for each suspicious app. N4- What permissions are requested by the malware(s)? Why it is suspicious? N5-Please provide a solution/s to quickly identify any suspicious API (please define your suspicious API according to your understanding) N6-What is the malware's home server URL and where is it located? Where, in the code, is/are stored the command server(s) URL(s) N7-What can you say about the communications model between the malware and its C&C server? N8-If encryption was used for the communication, which encryption algorithm was used? What was the key used? Explain how you found it. N9- Please draw a graph of the decrypted communication flow, found in the pcap, between the malware and the C&C N10-What personal informations were leaked during this incident? A special *secret* information was leaked, Explain how and what it was. N11-What particular techniques are used by the malware to harden analysis or to evade detection? What unusual behavior can be noticed? N12-Provide a detailed analysis of the malware behavior and features. N13-Please provide a method to block (or request permission from Android (similar to UAC concept)) when any suspicious call received from Android

表二：資訊安全實驗案例與 OWASP Mobile Top 10 各風險分類表

OWASP Mobile Top 10 (2014 v1.0)	Exploit-Me(E), HacmeBank(B), FourGoats(F), HerdFinancial(H), InsecureBank(P), Honey Challenges - Mobile Malware(N)
2014-M1	E2, B1, B2, B3, B4, B5, H2
2014-M2	E3, B6, F4, F5, P2, P3, P4
2014-M3	E1, E2, B2, B3, B4, B5, B9, F6, H6, P1, N1, N2
2014-M4	E4, E7, F3, P5, P10, N10
2014-M5	E8, B7, H1, N3, N4, N8
2014-M6	E5, B6, P11
2014-M7	B1, F1, P8, P9
2014-M8	F7, H4, P6
2014-M9	B2, B3, B4, F2, H3, N8
2014-M10	E6, B8, B10, B11, P13, N6
其它	F5, P7, P12, N5, N7, N9, N11, N12, N13

Android 應用程式之資訊安全滲透測試整合環境

由於要佈建 APP 資訊安全實驗案例的環境相當繁雜，因此 AppUse[45]於 2012 年 7 月釋出的免費 Android 應用程式滲透環境 AppSec Labs，該系統以 Linux Ubuntu 系統為基礎，集結了常用應用程式滲透相關工具外(例如 Android emulator, Eclipse development tool, required SDKs, apktool and dex2jar/JD-GUI decompliers, smali/baksmali disassemblers, Burp proxy, Wireshark 等)，亦包含特別為應用程式資訊安全測試環境所自製的 "hostile" Android ROM 與 emulator，透過 rootkit-like 技術，以 ReFrameworker Android runtime manipulator 進行應用程式的環境操控與執行期間的觀察。從安裝 apk 到 Android 裝置、進行反編譯、偵錯、操控執行環境等皆可透過其所設計的使用者介面輕易地完成，為目前 Android 應用程式之資訊安全實驗提供非常實用的整合型測試與學習環境。AppUse 已是 2014 年 BlackHat Conference 中行動裝置應用程式入侵的課程之一。

誘捕網路之行動裝置惡意程式挑戰實驗平台

Honeynet Project 於 2011 年 8 月於誘捕網路挑戰平台中釋出了行動裝置惡意程式分析之挑戰活動(Honeynet Project Challenges - Mobile Malware)提供相關資訊安全人員透過此挑戰設計學習如何進行行動裝置惡意程式之分析與鑑識技術的學習環境。跟上述資訊安全實驗案例平台不同的是，學習者必須從提供的真實案例足跡資訊中獨立分析惡意程式之行為。Honeynet Project 國際計畫始於 1999 年，參與國家與支會遍及全球，主要由多個模擬真實系統缺陷且不具營運價值的誘捕系統所構成的高互動式誘捕群集(Cluster of high-interactive honeypots)，用以誘捕駭客活動並藉以收集並分析各種威脅行為與攻擊手法進行防護之研究。該案例之挑戰主要提供兩個透過誘捕系統所採集之行動裝置真實惡意程式入侵案例資訊，包含一個已被破壞之行動裝置系統映像檔“data.bin”以及惡意程式通訊被捕抓的“traffic.pcap”檔案，透過這兩個足跡資訊來分析與鑑識並回答該挑戰所要

求的十三個問題，問題必須了解完整攻擊事件，並透過分析觀察回答問題與敘述研究方法，包含受害端的詳細資訊與被盜取之資訊、識別惡意程式與所使用之規避技術、惡意程式通訊行為與加密機制、惡意來源伺服器追蹤、以及惡意程式與命令控制伺服器(C&C)之通訊模式等。Honeynet Project Challenges 不僅可以讓我們學習到行動裝置惡意程式之惡意行為案例設計外，亦可從分析的過程了解惡意程式行為與鑑識之技術。

伍、Android 資訊安全實驗案例探討

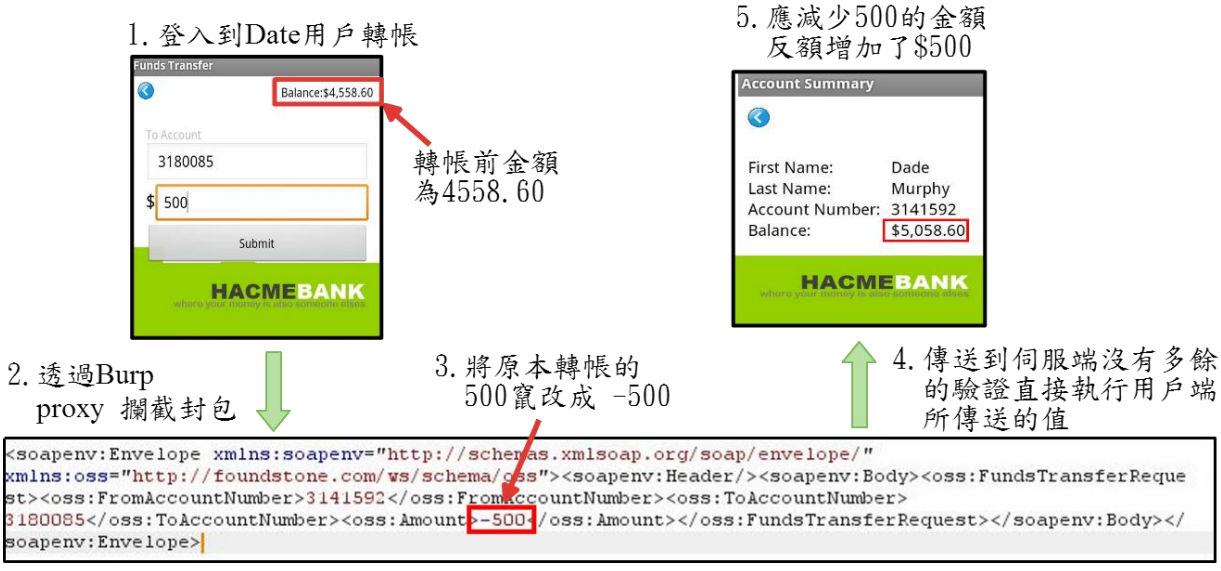
2014-M1: 伺服端缺乏完善的控管之實驗案例

我們以 HacmeBank - Lesson 1: SQL Injection 行動網路銀行之查詢用戶交易記錄為例，當用戶端傳送封包到伺服端資料庫系統，而資料庫系統沒有過濾或驗證用戶端所傳送的要求，因此透過 Burp suite proxy 攔截與竄改封包，如穿插惡意之 SQL injection 語法(xyz ' OR '1'='1)，藉由此方法取得伺服端之數據。

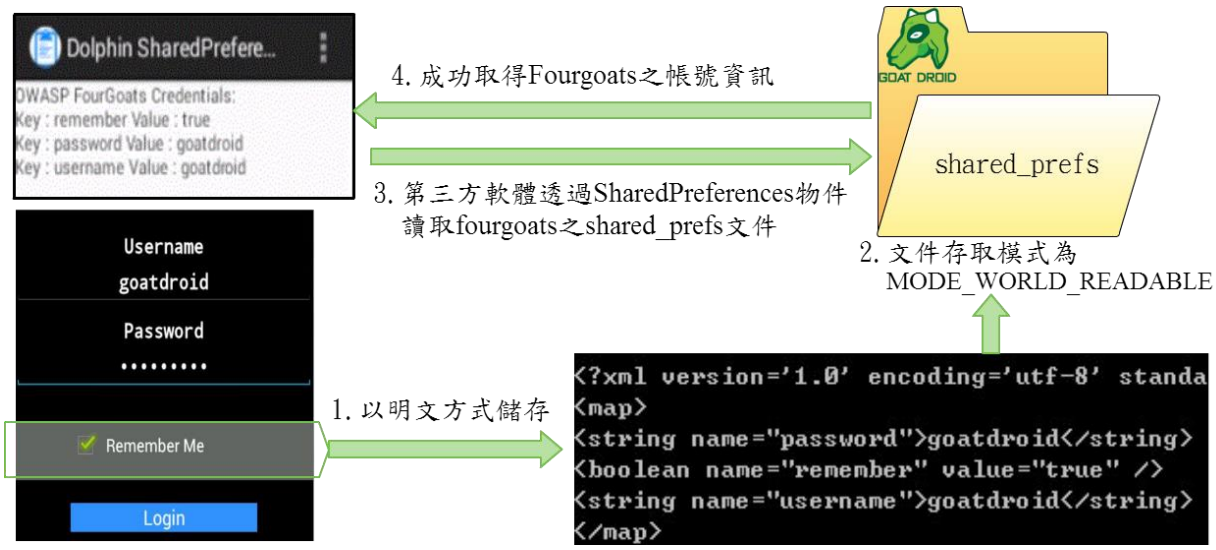
另外，我們再以 HacmeBank - Lesson 5: Application logic Bypass 案例為例，如圖二所示，由於轉帳金額僅在用戶端判斷與驗證，且伺服端邏輯層之判斷不夠嚴謹，如沒有完善的驗證而直接執行用戶端所傳送的資料，致使可以透過 Burp 攔截轉帳封包且竄改轉帳金額為負數，其存款理應減少之金額經竄改封包後反而增加了存款。

2014-M2: 儲存的資料無安全防護之實驗案例

儲存於行動裝置中的資料皆會受沙箱的保護，以避免其他應用程式濫用其他應用程式的資料。由於行動裝置內建之儲存空間有限，大部分的使用者皆會內接 SD card 來增



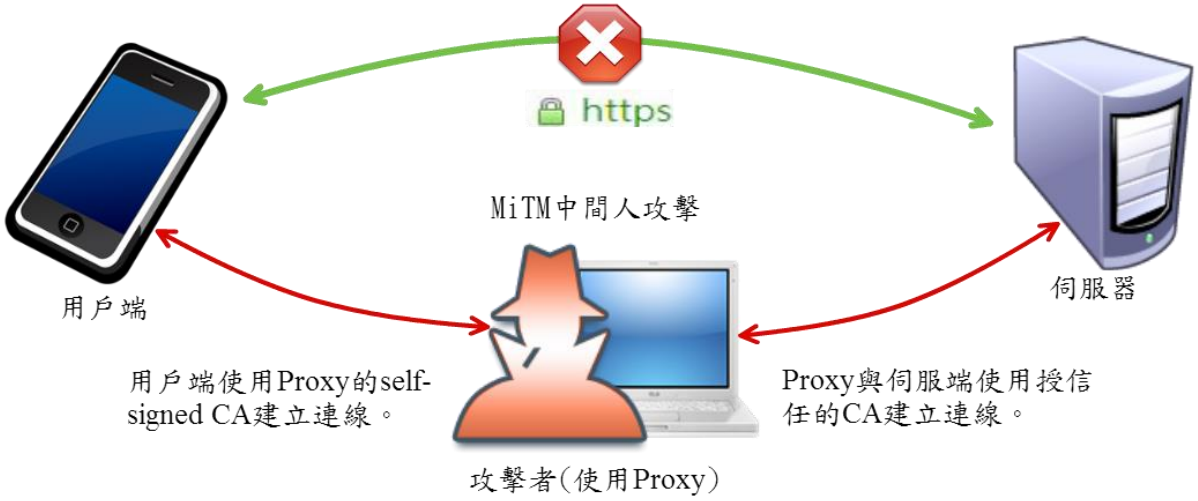
圖二、HacmeBank - Lesson 5: Application logic Bypass



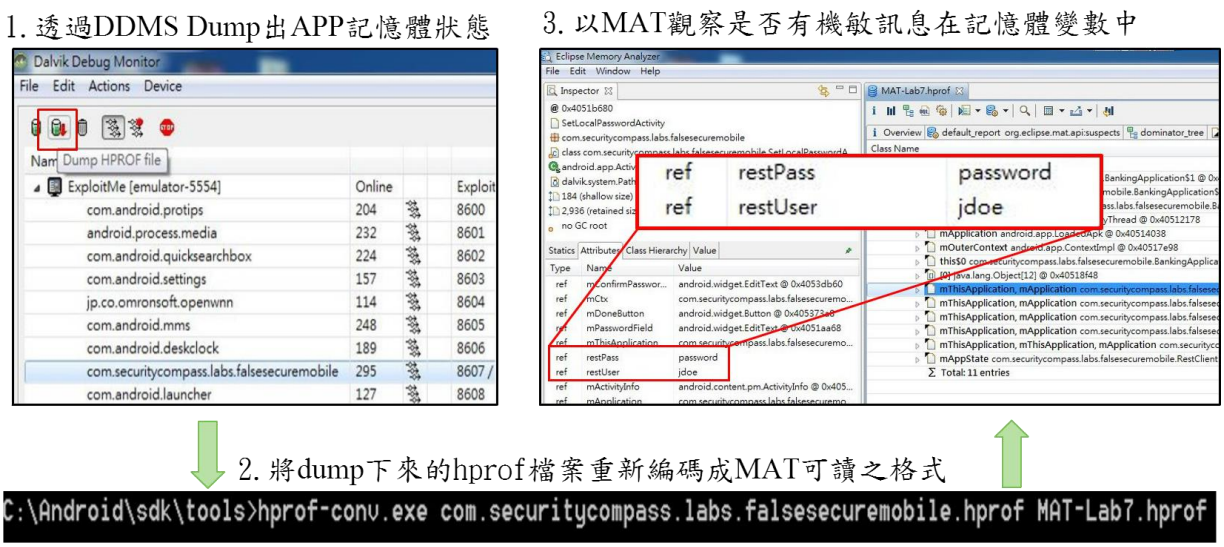
圖三、Fourgoats-Insecure Data 示意圖

加儲存容量，然而 SD card 是可抽拔的，當 SD Card 被拔出於其他裝置讀取時，除了沒有受到沙箱的保護，其原有設定之權限也會改變。以 EMMAL - Lab 3: Insecure File Storage 為例，可以發現儲存於 SD card 的資料權限是允許所有人都可以讀取的。

另外，我們以 Fourgoats - Insecure Data Storage 為例來說明明文儲存機敏資訊的風險，如圖三所示，因為案例之漏洞設計於在於應用程式記住帳號(Remember Me)之功能，此功能會將帳密儲存於裝置中，且存取模式有設定“MODE_WORLD_READABLE”，及代表允許其他應用程式讀取該文件，亦即第三方 APP 可透過 Fourgoats 的物件即可讀取該文件來取得帳號密碼，也可透過 adb 進入 Andorid shell 模式存取此 APP 之目錄文件，可發現帳號密碼皆未採用加防護直接儲存於行動裝置中。



圖四、HacmeBank - Lession 10 中間人 HTTPS 攻擊示意圖



圖五、EMMAL- Lab 7: Memory Protection

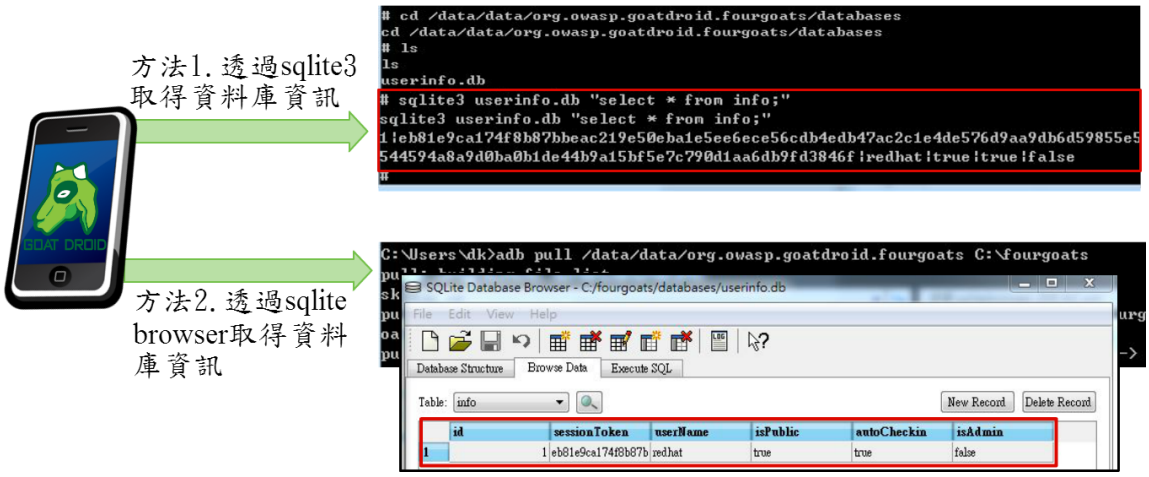
2014-M3: 傳輸層無足夠之防護之實驗案例

我們以 EMMAL- Lab 1: Secure Connection 來說明未加密明文傳輸之風險，因為案例之漏洞設計在於應用程式所輸入之帳號密碼傳輸至伺服端驗證端時沒有加密保護，因此可透過 tcpdump 搭配 WireShark 或 Proxy 來進行網路傳輸封包錄製分析與攔截，即可明確取得使用者端送出給伺服端認證的帳號與密碼。圖二所示範之案例亦為無加密以明文傳輸之案例。然當用戶端與伺服端使用 HTTPS 進行加密傳輸，亦可能遭受中間人攻擊。如 HacmeBank - Lesson 10 : Defeating SSL Certification Validation 案例於圖四所示，由於用戶端沒有強制確認 CA 憑證，因此當用戶端與伺服端進行 HTTPS 連線時，中間人攻擊可以以 Proxy 的 self-signed CA 與用戶端進行連線，在另與伺服端已受信任憑證進行連線，經由 Proxy 協助橋接雙邊之連線，但由於用戶端的憑證與加密金鑰是由 Proxy 發出的，因此不僅可以擷取用戶端送出的封包外，亦可直接進行封包解密。

2014-M4: 快取或暫存資料被竊取之實驗案例

此風險 EMMAL - Lab 7: Memory Protection 案例來探討如何透過 DDMS 來傾印應用程式執行期間之記憶體以取得有用之資訊。如圖五所示，首先從 DDMS 的介面中選擇 com.securitycompass.android.base 後，透過 dump hprof 功能傾印記憶體，因為 Dalvik VM 的關係，所以 HPROF dumps 並非正確的格式，所以我們在經由 Android SDK 提供的 hprof-conv 工具將 Android HPROF dumps 進行轉換，接著載入至 Eclipse 的 memory analyzer tool (MAT)進行分析，觀察應用程式存放於記憶體的變數，我們可以從中發現有變數包含了登入銀行的帳號與密碼。

2014-M5: 不嚴謹的授權機制與身份認證之實驗案例



圖六、FourGoat - Poor Authentication 之從 sqlite 取得資訊

本風險以 FourGoats - Poor Authentication 為案例來說明持續性認證之風險，該案例設計之漏洞在於 APP 的 Remember Me 功能中包含自動登入，如圖三所示，而帳號與連線權杖皆儲存於裝置內的 SQLite 資料庫中，如圖六所示，當藉由 adb shell 進入行動裝置 shell 互動模式來存取檔案系統找出.db 的資料庫檔，再透過 adb pull 複製出資料庫檔到 PC 再以 SQLite DataBase Browser 查詢資料庫內容，或使用 Android 提供的 sqlite3 指令進行 SQL 資料庫語法查詢。查詢結果中可以發現 Fourgoats 的 userinfo.db 資料庫檔案中存放使用者帳號與連線權杖等資訊，這些資訊可被有心人士進行連線劫持等惡意行為。

2014-M6: 加密演算法不嚴謹或被破解之實驗案例

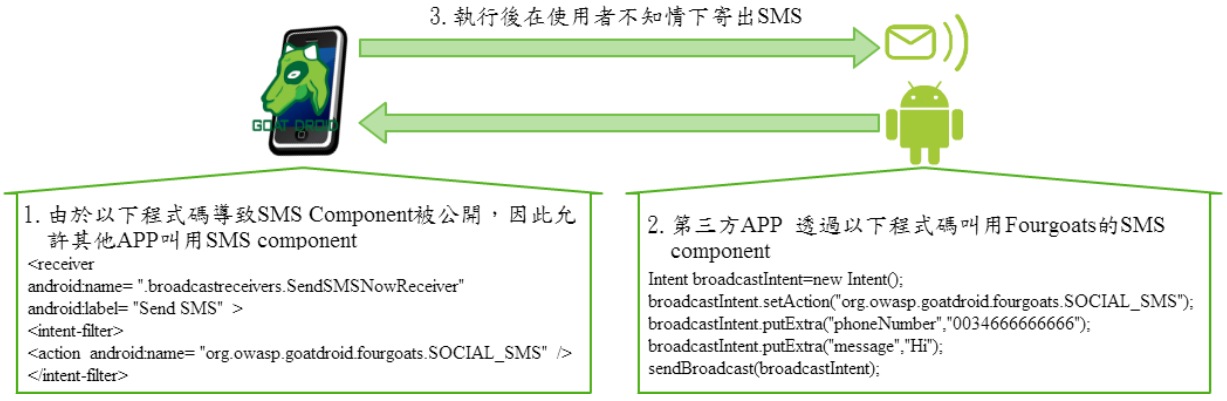
我們以 HacmeBank - Lesson 6: Insecure Data Storage 為案例來說明不嚴謹之加密演算法，首先透過 adb 進入 Android shell 模式找出 APP 儲存帳密之目錄檔案，取出已採用 Base64 編碼之帳密，再透過 Burp decoder 解碼即可將 Base64 編碼之帳密轉換成明文資訊。

接著再以 EMMAL-Lab6:Advanced Encryption 為案例來說明金鑰管理不當之風險，因為此案例主要使用預共用金鑰模式(Pre-shared key)對認證資訊進行加密，但這並不是好的加密方式。當應用程式載入並設置好設定檔後，接著觀察 xml 設定檔，可以發現認證資訊已經是加密的。然而我們可以透過 apktook 將 apk 檔案反編譯成.smali 格式，接者搜尋全部的.smali 檔案或從檔名中去觀察，最後即可從 CryptoTool.smali 檔案中發現存放解密金鑰，該金鑰可被取代取代成惡意人士的金鑰或透過該金鑰進行解密。

2014-M7: 用戶端注入攻擊變造之實驗案例

由於上述探討的所有資訊安全實驗平台中皆缺乏適合的用戶端注入攻擊案例，因此我們特別以 CVE-2011-2357[46]所暴露之漏洞為案例來探討如何 Android 2.3.4 與 3.1 版本之沙箱漏洞，致使可讓第三方 APP 進行 XSS 攻擊來取得瀏覽器認證之 cookie 資訊。如

圖七、CVE-2011-2357 Andorid 沙箱漏洞與瀏覽器 XSS 注入攻擊示意圖



圖八、FourGoats - Intent Spoofing 案例之漏洞與利用示意圖

圖七所示，Android 系統中的瀏覽器所存放之資訊是受沙箱保護的狀態，但此漏洞會致使 FLAG_ACTIVITY_BROUGHT_TO_FRONT 不被設定而可以透過 JavaScript 例如 “javascript:alert(document.cookie)”在瀏覽器目前所在頁面盜取並顯示 cookie 資訊。漏洞利用方法可以是開啟分頁數使其達到瀏覽器的上限後再開啟執行 JavaScrip 的分頁，或連續在短時間內開啟兩個網站分頁來達到同樣目的。第二種利用是因為此 Android 版本在連續開兩個分頁時，Android 的 Intent 處理機制有瑕疵不會設置上述標記 (正常情況下開啟網頁時會觸發 Intent 在新分頁顯示網頁內容)，因此可在開啟第二個分頁注入 JavaScript。

2014-M8: 針對不受信任的輸入之不當資安處置之實驗案例

我們以 FourGoats - Insecure IPC 為案例來說明行程間通訊(IPC)之不當行為。因為此

圖九、合法 APP 植入惡意 SMS 功能

案例 APP 具有收發 SMS 之權限的而 APP 的 Component 權限 android:exported 被設定為 true 且設定 SMS 的 intent-filter 沒有設定為 false，因此當其他 APP 或惡意行為之程序發出要求寄出 SMS 的 Intent 時 FourGoats APP 都會接收並發送 SMS。

2014-M9: 不當的連線會話管理之實驗案例

此風險我們以 HacmeBank- Lesson 2,3,4: Unauthorized Read, Write, and Execute 三個綜合案例來探討該行動銀行 APP 因無處理連線會話之身分轉換處理機制所造成之問題。此案例的身分驗證僅在用戶端進行驗證且傳輸至伺服端無進行加密，因此可透過 Burp proxy 操控連線會話進行認證身分轉換與劫持，例如進行竄改封包內的帳號資訊以取得別人之帳戶資料、竄改封包的使用者名稱進行認證身分轉換、竄改封包中匯款的目的帳號。

2014-M10: 應用程式缺乏二進位防護之實驗案例

APP 缺乏二進會防護以 HacmeBank Lesson 11-Defeating Authentication 為案例來說明所造成之風險。因為此案例之 apk 未採用二進位防護，因此可透過反組譯工具(apktool 等)來取得.smali 格式之程式碼，藉此觀察程式碼是否有程式上的疏失可利用(如身份驗證之行為或加入惡意代碼)，接著找出程式碼中的身份驗證之功能，將其修改成無效之身分驗證，最後重新組譯回 apk 安裝檔並安裝於行動裝置，如此該 APP 之登入系統只需使用合法的帳號但使用任意密碼即可通過認證機制登入銀行系統。

更進一步的我們另外示範如何將惡意代碼植入於合法 APP 中。如圖九所示，首先下載一個合法之日曆 APP，並透過反編譯與反組譯後(或從原始碼)於 APP 中加入 SMS 權限(“android.permission.SEND_SMS”和”android.permission.RECRIVE_SMS”)後，並再加入無須使用者確認即自動寄出 SMS 訊息之程式碼，最後組譯回 apk 並進行簽章。當使用者安裝並執行該 APP 時，惡意代碼將在使用者不知道的情況自動透過 SMS 傳送簡訊，例如可濫用該盜取的行動裝置 SMS 或進行購物等行為。

六、結論

本論文以資訊安全實驗來探討 OWASP 所提出之行動裝置應用十大風險，藉以透過實務案例來了解每個風險所帶來的風險與因應對策、弱點設計、與漏洞利用，除可有效提升 APP 開發者撰寫更安全的應用程式外，亦能增加使用者對於行動服務資訊安全的防護知識。綜整來說，新的威脅大部分皆來自行動應用的用戶端應用程式，尤其是應用程式間的資料盜取行為，而伺服端之威脅與傳統 PC 沒有太大的差異。另從各實驗案例與 OMT10 風險分類表(表二)可以看出，大量的案例凸顯了資料傳輸沒有保護、資料儲存不當被盜取、與 APP 編譯之二進位碼被破解之弱點，在 OMT10 中的 2014-M6、2014-M7、與 2014-M8 的案例相對較少。透過 Proxy 與逆向工程工具與技術可進行大部分的滲透分析。OMT10 中每個風險包含各種不同手法之攻擊，例如 2014-M8 的 Client Side Injection 中的 APP 弱點可被 Local File Inclusion、Intent Injection、與 Intent Fuzzing 等進行弱點利用，然而目前的行動裝置應用程式之資訊安全實驗案例涵蓋範圍不足，相關資訊安全工具與資源依然相當缺乏，也沒有一套完善的案例實驗環境與工具之整合系統，這些都是未來可以努力與研究的方向。

[誌謝]

本論文特別感謝本研究室團隊成員陳書正、黃冠龍、徐丞謙、張念祖、劉興銘、林聖元、宋皓榮研究生協助收集與測試行動裝置之資訊安全實驗案例。

參考文獻

[1] N. Ameya, P. Tomas, A. Mohammad, and Y. Kang, “Android mobile platform security and malware survey,” *IJRET: Internalational Journal of Research in Engineering and Technology*, vol. 2(11), pp. 764-774, 2013.

[2] M. Sonal and R. S. Sonar, “A Survey on mobile malware: a war without end,” *International Journal of Computer Science and Business Informatics*, vol. 9, no. 1, pp. 23-35, 2014.

[3] M. Tiwari, A. Srivastava and N. Gupta, “Review on Android and smartphone security,” *Research Journal of Computer and Information Technology Sciences*, vol. 1(6), pp. 12-19, 2013.

[4] Y. Rowland, “GINMASTER: A case study in android malware,” *Virus Bulletin Conference*, Oct 2013.

[5] http://www.appbrain.com/stats/number-of-android-apps (2014/3/13)

[6] http://www.forbes.com/sites/ericsavitz/2012/10/23/gartner-top-10-strategic-technology-trends-for-2013 (2012/10/23)

[7] http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-evolved-threats-in-a-post-pc-world.pdf (2012)
[8] http://www.juniper.net/us/en/local/pdf/additional-resources/3rd-jnpr-mobile-threats-report-exec-summary.pdf (2013)
[9] http://www.kaspersky.co.uk/about/news/virus/2012/threat_of_android_malware_skyrockets_into_2012 (2012/3/2)
[10] D. S. Alberts, J. J. Garstka and F. P. Stein, "Network centric warfare: developing and leveraging information superiority," *Washington DC: DoD Command and Control Research Program*, 1999.
[11] 王慧強,賴積保,朱亮與梁穎, "網絡態勢感知系統研究綜述," *計算機科學*, 2006.
[12] 景旭,唐晶磊與韓永國, "基於信息對抗的網絡集成防禦系統,", *微計算機信息*, 2006.
[13] 盧昱, "協同式網絡對抗," *北京:國防工業出版社*, 2003.
[14] http://wargame.cs.nctu.edu.tw/
[15] http://csep.mgt.ncu.edu.tw/
[16] http://securityoverride.org/challenges/index.php
[17] http://www.root-me.org/?lang=en
[18] https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project
[19] OWASP Mobile Security Project - Top Ten Mobile Risks https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
[20] http://securitycompass.github.io/AndroidLabs/
[21] https://www.honeynet.org/node/751
[22] https://www.owasp.org/index.php/Projects/OWASP_GoatDroid_Project
[23] http://www.mcafee.com/tw/downloads/free-tools/hacme-bank-android.aspx
[24] http://www.paladion.net/downloadapp.html
[25] http://developer.android.com/sdk/index.html
[26] http://www.busybox.net/
[27] http://sqlitebrowser.sourceforge.net/
[28] https://developer.android.com/tools/sdk/ndk/index.html
[29] http://www.android-x86.org
[30] https://www.virtualbox.org/
[31] https://code.google.com/p/android-apktool/
[32] https://code.google.com/p/dex2jar/
[33] http://jd.benow.ca/
[34] http://developer.android.com/tools/debugging/ddms.html
[35] https://www.eclipse.org/mat/

[36] http://portswigger.net/burp/
[37] https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
[37] https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
[39] http://www.charlesproxy.com/
[40] https://www.owasp.org/index.php/Main_Page
[41] http://www.project-rainbowcrack.com/
[42] https://www.owasp.org/index.php/Mobile_Top_10_2014-M10 (2014)
[43] http://www.arxan.com/
[44] http://flask.pocoo.org/
[45] https://appsec-labs.com/AppUse
[46] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2357 (2011)

[作者簡介]

許振銘博士在 2003/8~2007/6 年畢業於元智大學資訊工程博士班，2007/8~迄今任職於健行科技大學資訊工程系所助理教授並兼任電子計算機中心系統開發組組長至今。目前除為 EC-Council ECSP, CEH, CHFI, ECSA 認證講師外，亦為 Red Hat Inc. RHCE, RHCVA 認證講師與考官。主要研究興趣與實務專長為 Linux 系統滲透技術、行動裝置應用程式滲透、駭客手法分析與實務、巨量資料分析與探勘、雲端與虛擬化架構與技術、企業級 Linux 系統資訊安全強化。

許登凱目前就讀於健行科技大學資訊工程碩士班，主要之研究領域為雲端 IaaS 系統架構、弱點系統資訊安全實驗案例設計、與行動裝置資訊安全實驗整合環境。